目　　录

第1篇　语　言　篇

第1篇 语 言 篇

本篇的内容设计为适合大学二年级上学期的水平。这时学生已有了一定的计算机操作技能，同时又有矩阵运算的知识。这样，学生在学习本书的 1，2，3 章将不会有多少困难。我们制作的四小时录像带主要就针对这个部分。没有录像带的读者，只要有本书的光盘，也可以在计算机上复现录像中的所有屏幕画面，很容易对照自学。

MATLAB 是一种与数学水平密切相关的算法语言，第 4 章中介绍的内容需要较多的高等数学知识，要随着年级的增加才能逐渐深入掌握这些内容。在录像带中这部分占 20 分钟，读者在可根据自己的数学程度进行自学，并可与应用篇联系起来深入体会。

MATLAB 中还有一些大学本科中通常用不到的内容，但在毕业设计或今后的科研工程中可能有用，为了使本书具备手册的功能，这些内容用小字来叙述。同时，本书用小字列出了 MATLAB 基本部分的全部函数库，并配以索引，便于读者查找。这部分内容可以先跳过去，待需要时再看。

第 1 章　MATLAB 语言概述

1.1　MATLAB 语言的发展

MATLAB 是一种科学计算软件，适用于工程应用各领域的分析设计与复杂计算，它使用方便，输入简捷，运算高效且内容丰富，很容易由用户自行扩展。因此，当前已成为美国和其他发达国家大学教学和科学研究中最常用且必不可少的工具。

MATLAB 是由美国 Mathworks 公司于 1984 年正式推出的，到 1988 年有了 3.x（DOS）版本；1992 年推出 4.x（Windows）版本；1997 年推出 5.1（Windows）版本。以后又升级到 5.3（也称 R11）版本。2000 年下半年，Mathworks 公司推出了他们的最新产品 MATLAB 6.0（R12）试用版，并于 2001 年初推出了正式版。随着版本的升级，内容不断扩充，功能更加强大。另一方面对使用环境也提出了更高的要求。近几年来，Mathworks 公司在将 MATLAB 语言运用于系统仿真和实时运行等方面，取得了很多成绩，更扩大了它的应用前景。

其实，对于学习语法基础的读者来说，各版本的差别不大。考虑到国内的资源条件，本书将兼顾从 4.x 到 6.0 的各种版本。

MATLAB 是“矩阵实验室”（MATrix LABoratoy）的缩写，它是一种以矩阵运算为基础的交互式程序语言，着重针对科学计算、工程计算和绘图的需求。与其他计算机语言相比，其特点是简洁和智能化，适应科技专业人员的思维方式和书写习惯，使得编程和调试效率大大提高。它用解释方式工作，键入程序立即得出结果，人机交互性能好，为科技人员所乐于接受。特别是它可适应多种平台，并且随计算机硬、软件的更新而及时升级。因此，MATLAB 语言在国外的大学工学院中，特别是数值计算用得最频繁的电子信息类学科中，已成为每个学生都应掌握的工具了。它大大提高了课程教学、解题作业、分析研究的效率。学习掌握 MATLAB，也可以说是在科学计算工具上与国际接轨。

MATLAB 语言比较好学，因为它只有一种数据类型、一种标准的输入输出语句，不用“指针”、不需编译，比其他语言少了很多内容。听三四个小时课，上机练几个小时，就可入门了。以后自学也十分方便，通过它的演示（demo）和帮助（help）命令，人们可以方便地在线学习各种函数的用法及其内涵。

MATLAB 语言的难点是函数较多，仅基本部分就有 700 多个，其中常用的有二三百个，要尽量多记少查，才能提高编程效率，这也是终生受益的。

1.2　MATLAB 语言的特点

MATLAB 语言有以下特点。

1．起点高

（1）每个变量代表一个矩阵，从 MATLAB 名字的来源可知，它以矩阵运算见长。当前的科学计算中，几乎无处不用矩阵运算，这使它的优势得到了充分的体现。在 MATLAB 中，每个变量代表一个矩阵，它可以有 $n\times m$ 个元素。

（2）每个元素都看做复数，这个特点在其他语言中是不多见的。

（3）所有的运算都对矩阵和复数有效，包括加、减、乘、除、函数运算等。

2．人机界面适合科技人员

（1）MATLAB 的语言规则与笔算式相似。MATLAB 的程序与科技人员的书写习惯相近，因此，易写易读，易于在科技人员之间交流。

（2）矩阵的行列数无需定义。要输入一个矩阵，用其他语言时必须先定义矩阵的阶数，而 MATLAB 则不必有阶数定义语句。输入数据的行列数就决定了它的阶数。

（3）键入算式立即得结果，无需编译。MATLAB 是以解释方式工作的，即它对每条语句解释后立即执行，若有错误也立即做出反应，便于编程者立即改正。这些都大大减轻了编程和调试的工作量。

3．强大而简易的做图功能

（1）能根据输入数据自动确定坐标绘图。

（2）能规定多种坐标系（极坐标系、对数坐标系等）。

（3）能绘制三维坐标中的曲线和曲面。

（4）可设置不同颜色、线型、视角等。

如果数据齐全，通常只需一条命令即可出图。

4．智能化程度高

（1）绘图时自动选择最佳坐标，大大方便了用户。

（2）做数值积分时自动按精度选择步长。

（3）自动检测和显示程序错误的能力强，易于调试。

5．功能丰富，可扩展性强

MATLAB 软件包括基本部分和专业扩展两大部分。

基本部分包括矩阵的运算和各种变换、代数和超越方程的求解、数据处理和傅立叶变换及数值积分等等。可以充分满足大学理工科学生的计算需要。本书将介绍这部分的主要内容。

扩展部分称为工具箱。它实际上是用 MATLAB 的基本语句编成的各种子程序集，用于解决某一方面的专门问题，或实现某一类的新算法。现在已经有控制系统、信号处理、图像处理、系统辨识、模糊集合、神经元网络及小波分析等工具箱，并且向公式推导、系统仿真和实时运行等领域发展。

MATLAB 的核心内容在于它的基本部分，所有的工具箱子程序都是用它的基本语句编写的，学好这部分是掌握 MATLAB 必不可少的基础。

1.3　MATLAB 的工作环境

不同版本的 MATLAB 要安装在不同的操作系统下，MATLAB 3.x 之前的版本用的是 DOS 操作系统，而 MATLAB 4.x 以后的版本都以 Windows 操作系统为基础。MATLAB 的工作环境主要由命令窗（Command Window）、若干个图形窗（Figure Window）、文本编辑窗（File Editor）和文件管理窗（File Manager）组成，MATLAB 6.0 还增设了几个视窗。各视窗之间的切换可用【Alt】+【Tab】双键，即先按下【Alt】不放，再按【Tab】键；也可用鼠标在 Windows 界面的底部图标上单击实现。本章重点介绍命令窗，其他视窗将在读者对 MATLAB 有初步认识后再作详细介绍。

1.3.1　命令窗

在 Windows 桌面上，双击 MATLAB 的图标，系统就会进入 MATLAB 的工作环境，首先出现 MATLAB 的标志图形，接着出现命令窗。MATLAB 4.2 的命令窗如图 1.1 所示；MATLAB 5.x 的命令窗如图 1.2 所示；MATLAB 6.0 的命令窗如图 1.3 所示。

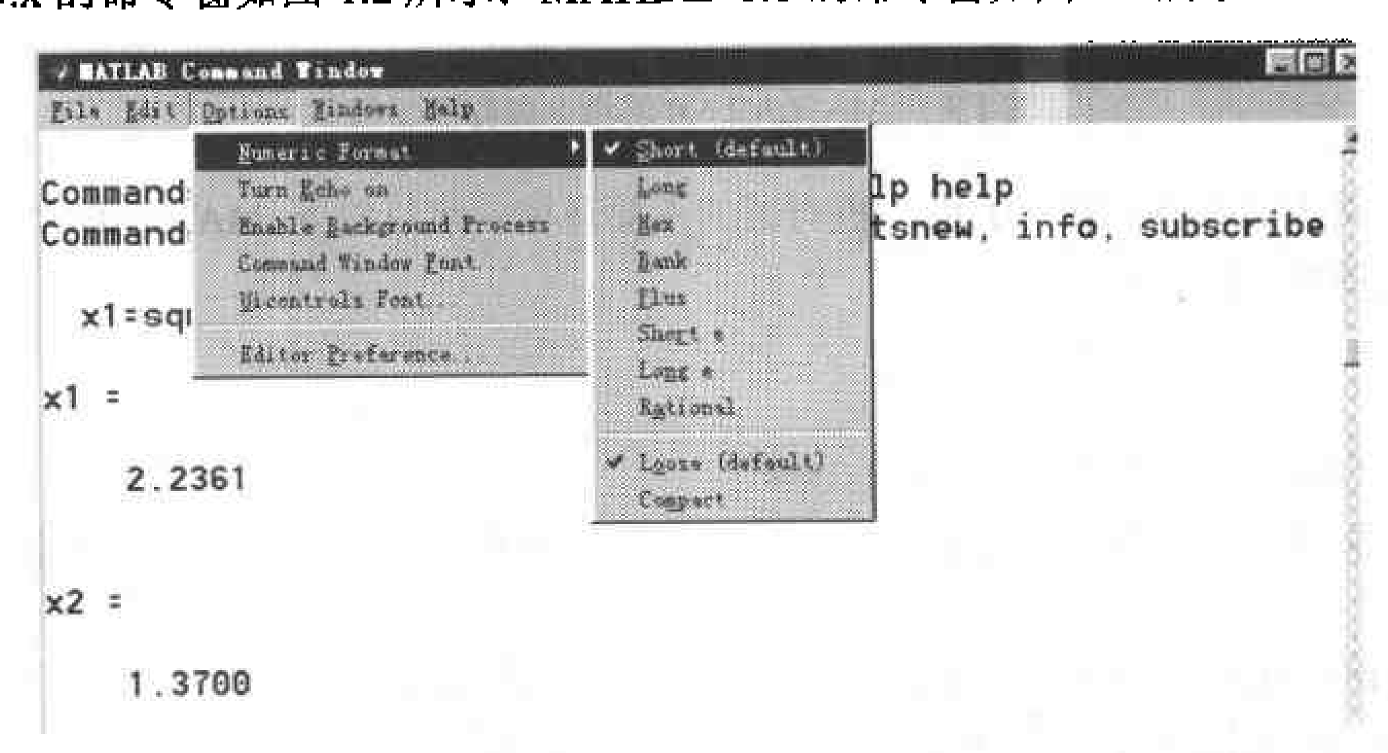


图 1.1　MATLAB 4.2c 的命令窗

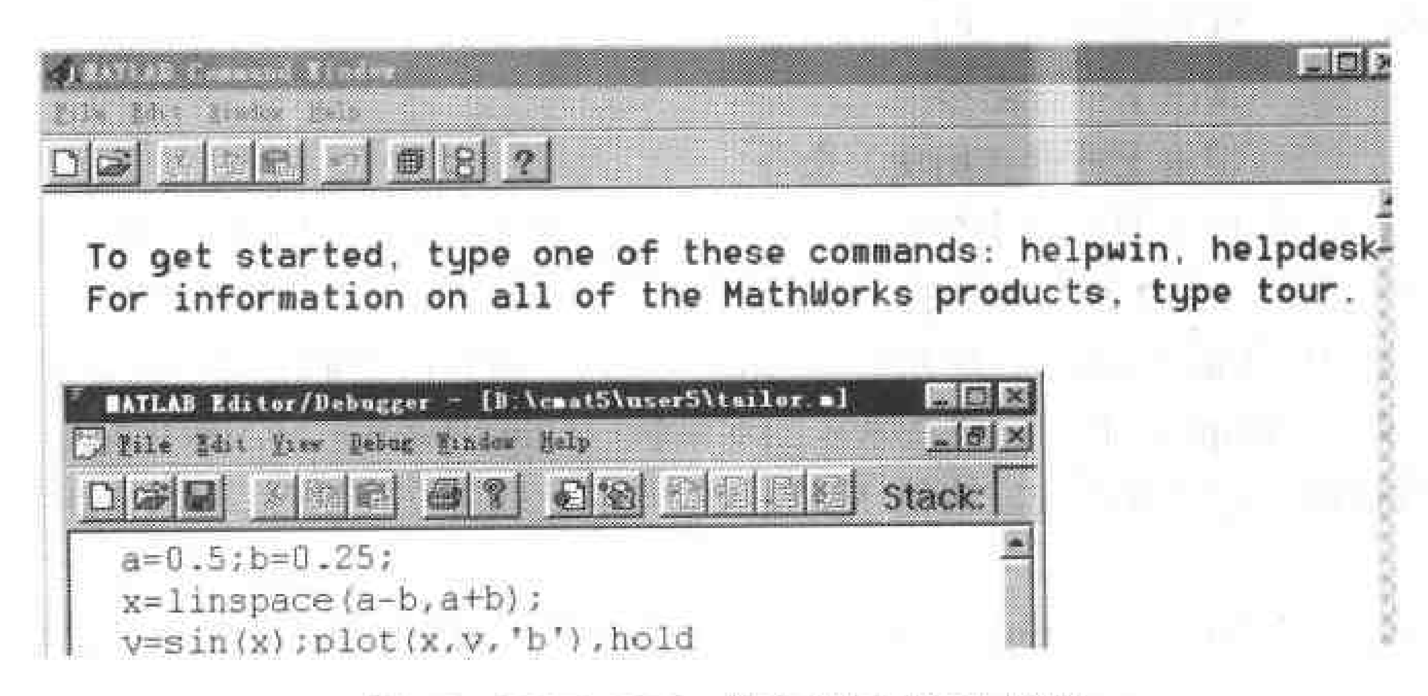


图 1.2　MATLAB 5.x 的命令窗界面上半部

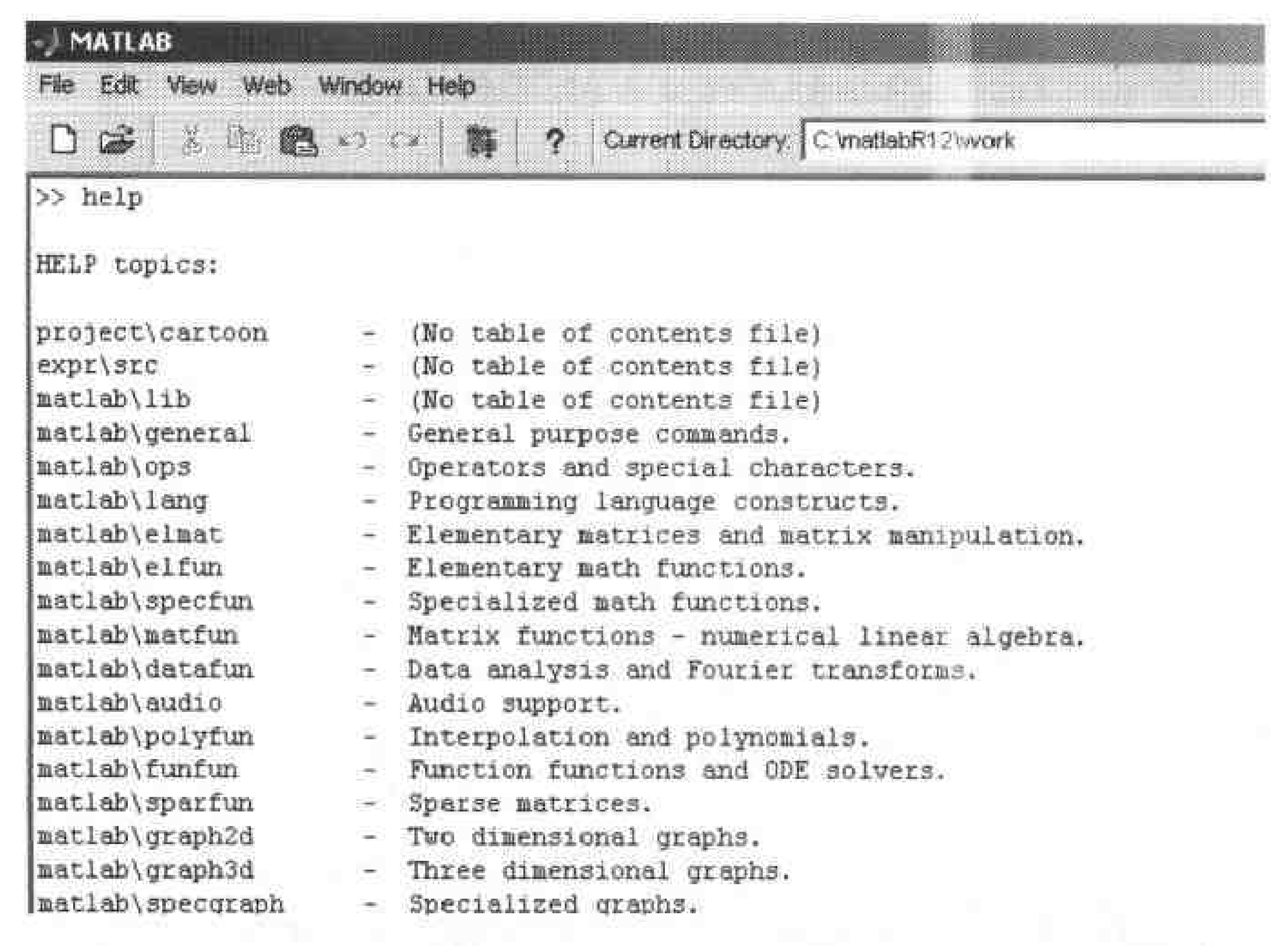


图 1.3　MATLAB 6.0 的命令窗界面顶部

命令窗是人们与 MATLAB 做人机对话的主要环境。可以键入 MATLAB 的各种命令并读出相应的结果。例如键入

```
x1=sqrt(5), x2=1.37, y=3/x2
```

答案为

```
x1 = 2.2361    x2 = 1.3700    y = 2.1898
```

说明　实际屏幕上的显示会占用 10 余行，为节省教材篇幅，删去了许多空行和空格，下同。

先对 MATLAB 4.2c 版本的命令窗做一些说明，因为它最简单，便于掌握。

● 命令窗编辑功能。键入和修改程序的方法与通常的文字处理相仿。先说明几个特殊的功能键。

➢ ESC　　恢复命令输入的空白状态

➢ ↓　　调出下一行命令

➢ ↑　　调出上一行（历史）命令

这几个功能在程序调试时十分有用。对于已执行过的命令，如要作些修改后重新执行，就可不必重新键入，用↑键调出原命令作修改即可。

● 主菜单中的编辑（Edit）项功能与 Word 相同。用它可以把屏幕上选定了的文字剪切（Cut）或复制（Copy）下来，放在系统的剪贴板（Clip Board）上，然后粘贴（Paste）到任一其他视窗的任何位置上去。这是 MATLAB 与其他软件交换文件、数据和图形的重要方法。

● 主菜单中的备选（Options）项功能。将它打开后又有 5 项子菜单，这里着重介绍其数字格式项（Numeric Format）。它的意义是选择显示数字的格式，将它打开就得到了 2 种显示格式和 8 种数字格式。

先讲显示格式，它分成稀疏（Loose）和紧凑（Compact）两种，默认（default）值为稀疏格式。若选择紧凑格式，同样执行上一条命令，则显示结果中将少了许多空行，使同一屏幕上可以显示更多的内容。MATLAB 存储和运算的数据只有一种格式，即 16 位十进制（二进制双精度）。但在屏幕上的显示却有 8 种格式，以便用户根据任务的需要进行选择。对同一个数的 8 种显示格式见表 1.1。

表 1.1　数字显示的 8 种格式

MATLAB 命令	显 示 形 式	说　　明
format long	35.83333333333334	16 位十进制数
format short e	35.833e+01	5 位十进制数加指数
format long e	35.83333333333334e+0	16 位十进制数加指数
format hex	4041aaaaaaaaaaab	16 位十六进制数
format bank	35.83	两位小数
format +	+	正、负或零
format rat	215 / 6	分数近似
format short（默认）	35.8333	两位整数，4 位小数

在 MATLAB 5.x 和 6.0 的主菜单中，取消了 Options 选项，因而只能利用最左列的程序命令来改变输出格式。对应于显示格式的命令为 format compact 和 format loose。如果只键入 format，则恢复默认设置，即数字格式为 short 而显示格式为 loose。显示格式也是 MATLAB 接收输入数据的格式。

● 主菜单的帮助(Help)项，有两个子菜单。一个是目录(Contents)，它给出了 MATLAB 基本部分的函数库名称及其包含的函数；另一个是索引(Index)，它按字母排序给出 MATLAB 各种函数和命令的用法，可以当做一个在线的说明书。在本书中将提供这两种查阅功能。与这个菜单项作用相同的是 help 命令，可能用得更多。

键入 help 即得到系统中已装入的函数库和工具箱（即子目录）名称，如图 1.3 所示。如果只装了 MATLAB 的基本部分，则屏幕上显示出的子目录名称如表 1.2 所示。如表 1.2 中所示将 MATLAB 4.2 和 MATLAB 5.x 版本的基本函数库分别列成两列。MATLAB 6.0 版本的子目录名称和 MATLAB 5.x 基本相同，只多了括号中的两个子目录(在表 1.2 中做了注明)。

➢ 键入 help 子目录名，如 help elfun，即得出 elfun 库中的全部函数名。

➢ 键入 help　函数名，如 help tan2，即得到 tan2 函数的意义及用法。

从 MATLAB 5.x 起，增加了 HTML 帮助功能。在联网条件下，或系统已安装了 HTML 帮助文件时，键入 doc 或单击菜单项 help 下的子菜单 Help Desk（html），即可实现联网帮助。

● 命令窗的右边有上下滚动条，它是用来翻动页面的，用它可以向前翻动命令窗中过去的显示内容。但保留的显示空间是有限的。能保留的记录通常不少于 10 个页面，这还要看页面中字符的多少而定。命令窗下设有左右滚动条，当输入命令或输出数据行超过 80 个字符时，可用它来左右翻动页面。

● 退出 MATLAB 有两种方法。一是键入 exit 或 quit；还有一种是用鼠标双击窗口左上角的小方块或单击右上角的×号。

表 1.2　MATLAB 基本部分的函数库

MATLAB 4 中的库名	库　内　容	MATLAB 5 和 MATLAB 6 中的库名	库 序 号	在本书中的章节
datafun	数据分析和傅立叶变换函数库	datafun (audio)	(a)	4.1 节表 4.3
sounds	声音处理函数库			MATLAB 6 又将声音库(audio)分出来
dde	客户机函数库	dde	(g)	3.3 节表 3.4
elfun	基本函数库	elfun	(c)	2.3 节表 2.7
specmat	特殊矩阵库	elmat	(d)	2.1 节表 2.1
elmat	初等矩阵库			
	时间和日期函数库	timefun	(w)	3.1 节表 3.2
funfun	函数功能和数值积分库	funfun	(e)	4.4 节表 4.5
general	通用命令库	general	(f)	3.1 节表 3.1
无	数据类型和结构库	datatypes	(b)	4.7 节表 4.10
graphics	通用图形函数库	graphics	(h)	2.5 节表 2.13
iofun	低层文件输入/输出库	iofun	(j)	3.3 节表 3.3
lang	语言结构库	lang	(k)	2.6 节表 2.17
matfun	矩阵函数和数值线性代数库	matfun	(m)	4.2 节表 4.4
ops	运算符和特殊字符库	ops	(n)	2.4 节表 2.9
plotxy	二维图形函数库	graph2d	(p)	2.5 节表 2.14
	特殊图形库	specgraph	(u)	2.6 节表 2.15
plotxyz	三维绘图和光照函数库	graph3d	(q)	2.5 节表 2.16
color	颜色和光照函数库			
polyfun	多项式和插值函数库	polyfun	(r)	4.3 节表 4.5
sparfun	稀疏矩阵函数库	sparfun	(s)	4.6 节表 4.8
specfun	特殊函数库	specfun	(t)	4.4 节表 4.6
strfun	字符串函数库	strfun	(v)	4.5 节表 4.7
无	图形用户界面工具函数库	Guitools	(x)	4.7 节表 4.9
无	版本控制函数库	(verctrl)	(z)	MATLAB 6.0 新增的函数库
demos	MATLAB 演示库	demos	(y)	未列出

现在看图 1.2 所示的 MATLAB 5.x 命令窗界面（可以先不去管在图中已打开的文本编辑窗口），它多了一条由 9 个图标组成的工具栏。左边两个按钮用以打开文本编辑器及程序文件；第 3 到第 6 个按钮是 Edit 菜单项中的快捷键，是用来直接进行命令窗编辑的。左数第 7 个按钮用来查看 MATLAB 工作空间内的变量；第 8 个按钮用来改变 MATLAB 的搜索路径，部分执行了文件管理的功能，它的用法将在第 3 章中介绍。

图 1.3 所示为 MATLAB 6.0 的命令窗菜单区。从外观上看，它增加了一个显示当前目录的信息区。在主菜单上增加了 Web 项，表明了它在联网功能上的加强。它的其他功能扩展主要反映在子菜单中，各子菜单项之下的菜单也有扩展。MATLAB 6.0 的命令窗菜单中增加的项目和功能，将在第 3 章中介绍。

1.3.2　图形窗

通常，只要执行了任一种绘图命令，就会自动产生图形窗。以后的绘图都在这一个图形窗中进行。如想再建一个或几个图形窗，则可键入 figure，MATLAB 会新建一个图形窗，并自动给它依次排序。如果要人为规定新图为图 3，则可键入 figure (3)。

1.3.3　文本编辑窗

MATLAB 程序编制有两种方式，一种称为行命令方式，这就是在命令窗中一行一行地输入程序，计算机每次对一行命令做出反应，像计算器那样。这只能编简单的程序，在入门时可以用这种方式。程序稍复杂一些，就应把程序写成一个有多行语句组成的文件，让 MATLAB 来执行这个文件。编写和修改这种文件程序就要用到文本编辑器。

MATLAB 4.x 的文本编辑器不太完善，可以借用 Windows 中的记事本（Notepad）来做文本编辑。MATLAB 5.x 和 MATLAB 6.0 的编辑器的操作很接近于 Word，比较完善。它自动用颜色区别程序的不同内容，容易直接发现错误，并且与程序调试相结合，应该尽量用这个编辑器。

1.3.4　文件管理窗

在较为复杂的程序中，特别是涉及与其他软件系统进行文件或数据交换时，需要用到文件管理系统。MATLAB 4.x 没有专门的文件管理器。MATLAB 5.x 和 MATLAB 6.0 已把文件管理功能集成到 MATLAB 命令窗中，使用更加方便。

后 3 种视窗的功能和用法将在第 3 章中介绍。

1.4　演 示 程 序

在命令窗中键入 demo，将出现 MATLAB 的演示窗。MATLAB 4.2 的演示窗有图文并茂的演示菜单，如图 1.4 所示。

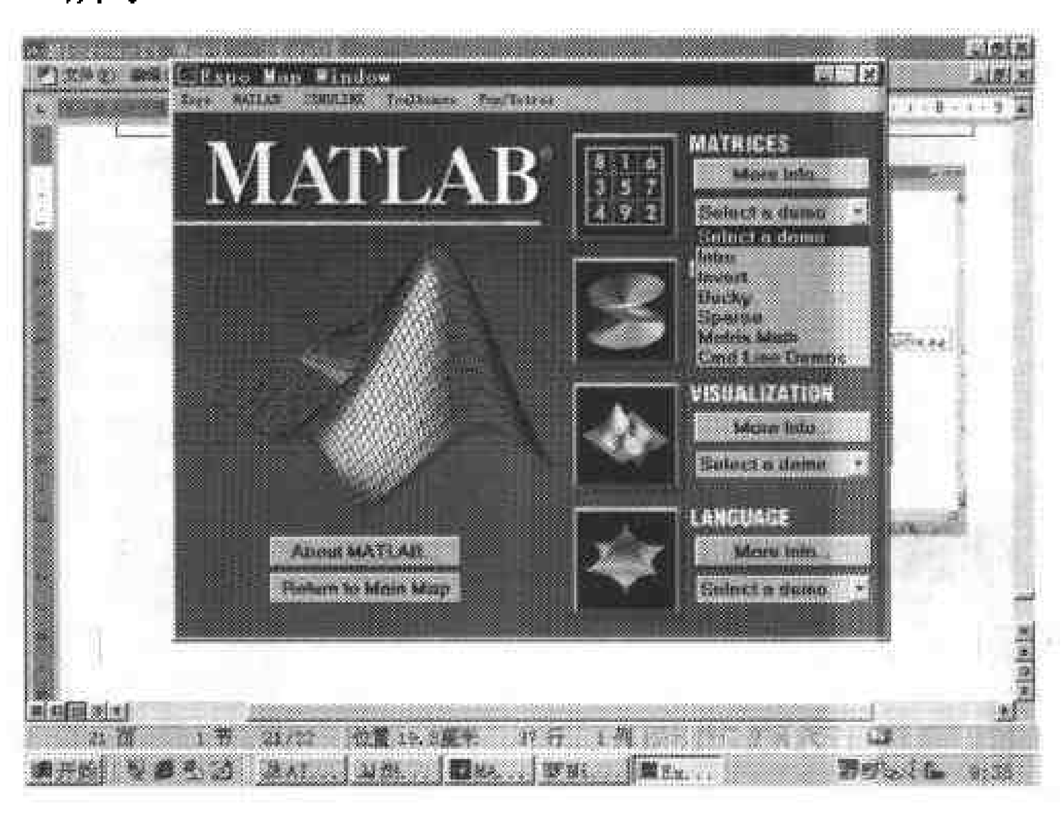


图 1.4　MATLAB 4.2 的演示界面

对于只装了基本部分的系统，不能演示 Simulink 和 Toolbox 的内容。只能选择 MATLAB 项，弹出下一层菜单，该菜单有以下 4 个选项。

- 矩阵运算（Matrices）　内有 6 个矩阵运算和变换的演示实例；
- 数值计算（Numerics）　内有 5 个数值求解和求积分的演示实例；
- 可视化（Visualization）　内有 8 个绘图、动画、声音的演示实例；
- 语言（Language）　内有 4 个设置坐标、颜色的演示实例；

MATLAB 5.x 和 MATLAB 6.0 的演示窗则全是英文文字菜单。如图 1.5 所示。读者可在计算机上自行试用。在演示实例时，通常画面的上半部是图形，而下半部则是相应的 MATLAB 程序语句。图 1.6 就显示了 z 为复数时 f(z) = z^3 的三维图形。读者还可以在界面上直接修改这些语句并重新执行。因此，演示程序也是一个很好的学习过程，可以作为对 MATLAB 功能的一次浏览。

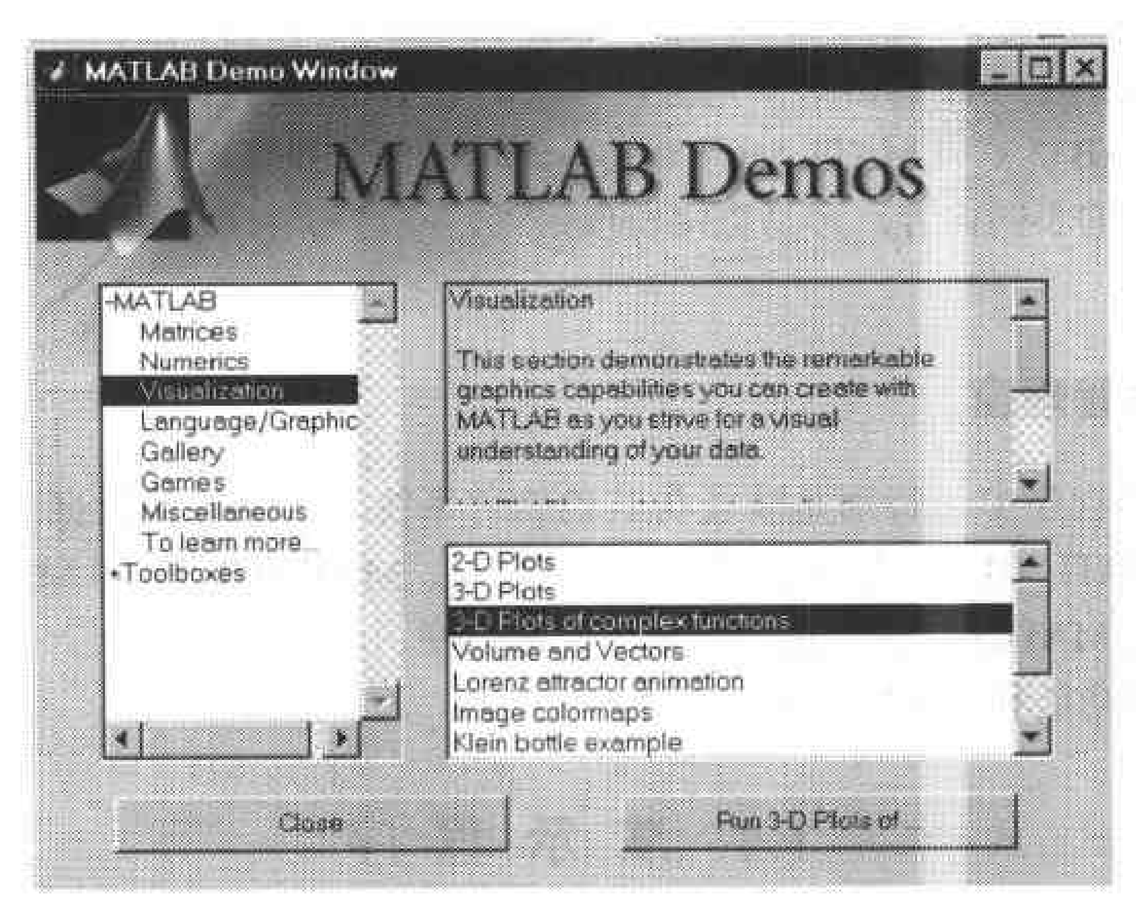


图 1.5　MATLAB 5.x 和 MATLAB 6.0 的演示窗

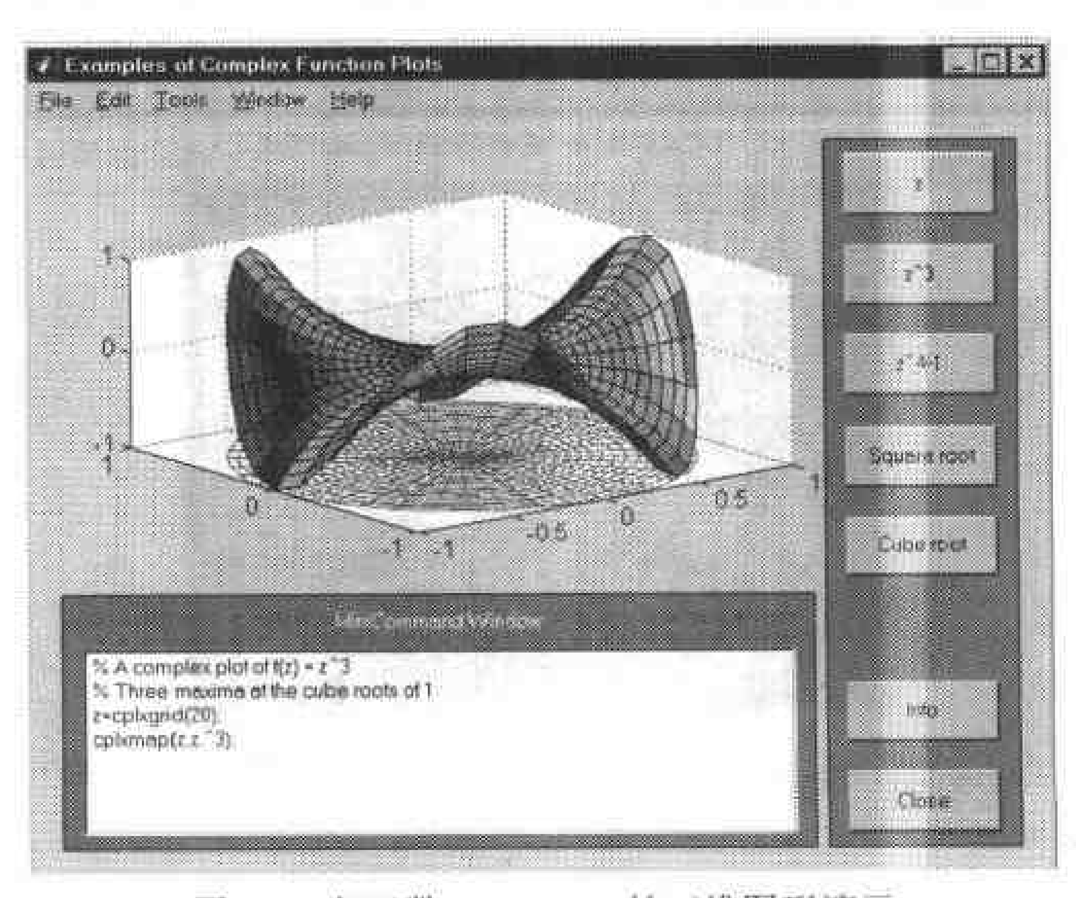


图 1.6　复函数 f(z) = z^3 的三维图形演示

第2章 基本语法

本章主要介绍MATLAB的基本语法。

2.1 变量及其赋值

2.1.1 标识符与数

标识符是标志变量名、常量名、函数名和文件名的字符串的总称。在MATLAB中，变量和常量的标识符最长允许19个字符；函数和文件名则通常不超过8个字符（受文件管理器的限制，但MATLAB 5.1的许多函数和文件名就超过了8个字符，因此，要求安装在能管理长文件名的操作系统中）。这些字符包括全部的英文字母（大小写共52个）、阿拉伯数字和下划线等符号。标识符中第1个字符必须是英文字母。MATLAB对大小写敏感（Case Sensitive），即它把A和a看做两个不同的字符，这是默认状态。如果要不区分大小写，应键入命令casesen off；若再键入casesen on，则恢复默认状态。

MATLAB内部只有一种数据格式，那就是双精度（即64位）二进制，对应于十进制16位有效数和±308次幂。MATLAB作运算和存储时都用双精度格式，这对绝大多数工程计算是足够的，许多情况下甚至过于“浪费”。在一些其他的算法语言中设有多种数据格式，如字符型（8位）、整数型（16位）、单精度型（32位）等，可节省内存和提高速度，但增加了编程的复杂性。MATLAB把简化编程作为其主要目标，省去了多种数据格式，但在运算速度和内存消耗方面付出代价。不过，现在计算机的时钟频率和内存容量都以几何级数迅速增长，MATLAB付出的代价容易得到弥补。虽然它的数据格式只有一种，但为了人机交互的友好方便，输出显示格式有8种，已在第1章中指出。

2.1.2 矩阵及其元素的赋值

赋值就是把数赋予代表常量或变量的标识符。MATLAB中的变量或常量都代表矩阵，标量应看做1×1阶的矩阵。赋值语句的一般形式为：

➢ 变量=表达式（或数）

例如输入语句

```
a = [1  2  3; 4  5  6; 7  8  9]
```

则显示结果为

```
a =  1  2  3
     4  5  6
     7  8  9
```

元素也可以用表达式代替，如输入 `x=[-1.3 sqrt(3) (1+2+3)/5*4]`

结果为 `x=-1.3000    1.7321    4.8000`

可以看出，矩阵的值放在方括号中，同一行中各元素之间以逗号或空格分开，不同的行则以分号隔开，语句的结尾可用回车符或逗号，此时会立即显示运算结果。如果不希望显示结果，就以分号结尾。此时运算仍然执行，只是不作显示。

变量的元素用圆括号“()”中的数字（也称为下标）来注明，一维矩阵（也称数组或向量）中的元素用一个下标表示，二维的矩阵可有两个下标数，以逗号分开。MATLAB 5.x 已扩展了三维和更高维的矩阵，可有三个或更多下标。用户可以单独给元素赋值，如：x(2)=1.7321，a(2，3)=6 等。如果赋值元素的下标超出了原来矩阵的大小，矩阵的行列会自动扩展。如：

```
x(5)=abs(x(1))
```

得

```
x =-1.3000    1.7321    4.8000     0     1.3000
```

又如键入 `a(4, 3)=6.5`

得

```
a =  1.0000    2.0000    3.0000
     4.0000    5.0000    6.0000
     7.0000    8.0000    9.0000
     0         0         6.5000
```

可见，跳空的元素 x(4)，a(4，1)，a(4，2)被自动赋值 0。这种自动扩展维数的功能只适用于赋值语句。在其他语句中若出现超维调用的情况，系统将给出出错提示。

给全行赋值，可用冒号。例如，给 a 的第 5 行赋值。

键入

```
a(5, : )=[5, 4, 3]
```

得

```
a =  1.0000      2.0000      3.0000
     4.0000      5.0000      6.0000
     7.0000      8.0000      9.0000
     0           0           6.5000
     5.0000      4.0000      3.0000
```

把 a 的第 2、4 行及第 1、3 列交点上的元素取出，构成一个新矩阵 b。

可键入

```
b = a([2, 4], [1, 3])
```

得

```
b =  4.0000      6.0000
     0           6.5000
```

要抽去 a 中的第 2、4、5 行，可利用空矩阵[ ]的概念。

键入

```
a([2, 4, 5], :)=[ ]
```

得

```
a = 1     2     3
    7     8     9
```

注意，“空矩阵”是指没有元素的矩阵。对任何一个矩阵赋值[ ]，就是使它的元素都消失掉。这完全不同于“零矩阵”，后者是元素存在，只是其数值为零而已。可以看出，空矩阵是使矩阵减缩时不可缺少的概念。

除“变量＝表达式（或数）”的标准赋值格式外，还可以不要等式左端而只剩下“表达式”。这有两种可能：（1）该表达式并不产生数字解，例如，产生图形或改变系统状态；（2）该表达式产生数字解，但不需保存它。此时，MATLAB 自动给出一个临时变量 ans，把右端的结果暂存在 ans 中。例如键入 `a/7`

得

```
ans =    0.1429    0.2857    0.4286
```

```
1.0000    1.1429    1.2857
```

2.1.3 复数

MATLAB 的每一个元素都可以是复数，实数是复数的特例。复数的虚数部分用 i 或 j 表示。这是在 MATLAB 启动时就在内部设定的。例如：

键入　c = 3+5.2i

得　c = 3.0000 + 5.2000i

对复数矩阵有两种赋值方法。

（1）将其元素逐个赋予复数，如：

键入　z=[1+2i, 3+4i;5+6i, 7+8i]

得
```
z =  1.0000 + 2.0000i    3.0000 + 4.0000i
     5.0000 + 6.0000i    7.0000 + 8.0000i
```

（2）将其实部和虚部矩阵分别赋值，如：

```
z = [1, 3;5, 7]+[2, 4;6, 8]*i
```

两种赋值方法得出同样的结果。注意，只有数字和 i 的乘积可省略乘号，在上述矩阵式中若省略乘号"*"，就会出错。另外，如果在前面程序中曾经给 i 或 j 赋过其他值，则 i，j 已经不是虚数符号，这些虚数赋值语句都不对了。此时应键入：

```
clear i, j
```

即把原赋的 i，j 清掉，然后再执行复数赋值语句。

MATLAB 中所有的运算符和函数都对复数有效。例如：

键入　f = sqrt(1+2i)

得　f = 1.2720 + 0.7862i

检验　f*f

```
ans =   1.0000 + 2.0000i
```

因此，复数的表达式同样也能作为赋值语句。再来看复数矩阵 z 的转置、共轭运算，运算符 ' 表示把矩阵作共轭转置，即把它的行列互换，同时，把各元素的虚部反号。函数 conj 则只把各元素的虚部反号，即只取共轭。所以，若求转置而不要共轭，就把 conj 和 ' 结合起来完成。

键入　w=z'(共轭转置)，u=conj(z)（共轭），v=conj(z)'（转置）

得
```
w =  1.0000 - 2.0000i     5.0000 - 6.0000i
     3.0000 - 4.0000i     7.0000 - 8.0000i
u =  1.0000 - 2.0000i     3.0000   4.0000i
     5.0000   6.0000i     7.0000 - 8.0000i
v =  1.0000 + 2.0000i     5.0000 + 6.0000i
     3.0000 + 4.0000i     7.0000 + 8.0000i
```

2.1.4 变量检查

在调试程序时，往往需要检查工作空间中的变量及其阶数。可用 who 或 whos 命令。

键入　who

得　Your variables are:

```
a       c       v       x1      z
ans     f       w       x2
b       u       x       y
```

这些是前面用过的变量，如果还需要知道它们的详细特征，可键入：

```
    whos
```

得

变量名	阶数	元素数	字节数	密度	复数
a	2 by 3	6	48	Full	No
ans	2 by 2	4	64	Full	Yes
b	2 by 2	4	32	Full	No
c	1 by 1	1	16	Full	Yes
...	...	...	...	...	...
z	2 by 2	4	64	Full	Yes

Grand total is 40 elements using 496 bytes(共 40 个元素，占 496 字节)

可以看出，每个实元素占 8 个字节，复元素则占 16 个字节。读者可自行解释其原因。

MATLAB 中实际上还有几个内定的变量，在变量检查时不显示。把它们列于表 2.1 的特殊变量栏中。这里着重介绍一下 Inf 和 NaN。

表 2.1　基本矩阵和矩阵运算(elmat) (d)

基本矩阵	zeros	全 0 矩阵（$m\times n$ 阶）	logspace	对数均分向量（$1\times n$ 维数组）
	ones	全 1 矩阵（$m\times n$ 阶）	Freqspace	频率特性的频率区间
	rand	随机数矩阵（$m\times n$ 阶）	meshgrid	画三维曲面时的 X，Y 网格
	randn	正态随机数矩阵($m\times n$ 阶)	meshdom**	在 MATLAB5.x 及以后版本中取消
	eye(n)	单位矩阵（方阵）	:	将元素按列取出排成一列
	Linspace	均分向量（$1\times n$ 维数组）		
特殊变量和函数	ans	最近的答案	inf	Infinity（无穷大）
	eps	浮点数相对精度	NaN	Not-a-Number（非数）
	realmax	最大浮点实数	flops	浮点运算次数
	realmin	最小浮点实数	computer	计算机类型
	pi	3.14159235358579	inputname *	输入变量名
	i，j	虚数单位	size	多维矩阵的各维长度
	length	一维矩阵的长度		
矩阵结构提取和变换	cat *	链接数组	diag	提取或建立对角阵
	fliplr	矩阵左右翻转	ind2sub *	把元素序号变为矩阵下标
	flipud	矩阵上下翻转	sub2ind *	把矩阵下标变为元素序号
	repmat	复制和排成矩阵	tril	取矩阵的左下三角部分
	reshape	维数重组(元素总数不变）	triu	取矩阵的右上三角部分
	Rot90	矩阵整体反时针旋转 90 度		
特殊矩阵	compan	Companion 矩阵	magic	魔方矩阵
	gallery	Higham 测试矩阵	pascal	Pascal 矩阵
	hadamard	Hadamard 矩阵	rosser	经典的对称特征值测试问题

续表

特殊矩阵	hankel	Hankel 矩阵	Toeplitz	Toeplitz 矩阵
	hilb	Hilbert 矩阵	vander	Vandermonde 矩阵
	invhilb	Hilbert 逆矩阵	wilkinson	Wilkinson's 特征值测试矩阵

注：* 表示 MATLAB 5.x 中增加的内容。

** 表示 MATLAB 5.x 及以后版本中取消的内容。以后的表中*和**的意义与此相同。

Inf（还有 -Inf）是无穷大，键入 1/0 就可得到它。NaN 是非数字（Not a Number）的缩写，由 0/0，0*Inf 或 Inf/Inf 而得。在其他语言中遇到上述非法运算时，系统就停止运算并退出。而 MATLAB 却不停止运算，仍给结果赋予 Inf 或 NaN，并继续把程序执行完。这有很大的好处，可以避免因为一个数据不好而破坏全局。出现 Inf 或 NaN 后，对它们做任何运算，结果仍为 Inf 或 NaN，这种运算规则称为 IEEE（电工和电子工程师协会）运算规则，是 IEEE 的一种标准。

2.1.5 基本赋值矩阵

为了方便给大量元素赋值，MATLAB 提供了一些基本矩阵。表 2.1 给出了最常用的一些。其用法可从下面的例子中看到。其中，魔方矩阵 magic(n)的特点是：其元素由 1 到 nn 的自然数组成；每行、每列及两对角线上的元素之和均等于$(n^3+n)/2$。单位矩阵 eye(n)是 n×n 阶的方阵，其对角线上的元素为 1，其余元素均等于 0。下例列出了表 2.1 中的 4 种矩阵。

```
键入      f1=ones(3, 2), f2=zeros(2, 3), f3=magic(3), f4=eye(2)
得全 1 矩阵      f1 = 1    1
                      1    1
                      1    1
  全 0 矩阵      f2 = 0    0    0
                      0    0    0
  魔方矩阵   f3 = 8    1    6
                  3    5    7
                  4    9    2
  单位矩阵   f4 = 1    0
                  0    1
```

线性分割函数 linspace(a, b, n)在 a 与 b 之间均匀地产生 n 个点值，形成 n 维向量。如：

```
键入    f5 = linspace(0, 1, 5)
得      f5 =  0     0.2500     0.5000     0.7500     1.0000
```

大矩阵可由若干个小矩阵组成，但必须其行列数正确，恰好填满全部元素。如：

```
键入    fb1=[f1, f3; f4, f2]
得      fb1 =    1    1    8    1    6
                 1    1    3    5    7
                 1    1    4    9    2
                 1    0    0    0    0
                 0    1    0    0    0
```

可再由它与 f5 组成一个更大的矩阵。

键入　fb2=[fb1;f5]

得
```
fb2 =   1.0000   1.0000   8.0000   1.0000   6.0000
        1.0000   1.0000   3.0000   5.0000   7.0000
        1.0000   1.0000   4.0000   9.0000   2.0000
        1.0000   0        0        0        0
        0        1.0000   0        0        0
        0        0.2500   0.5000   0.7500   1.0000
```

可以看出 fb1 和 fb2 显示的数据不同，fb1 的元素都是整数，而 fb2 则都是带小数的。其原因是 MATLAB 要求一个矩阵中所有元素用同一显示格式。因为 f5 中元素含小数，所以所有的元素都得用小数格式来显示（0 元素除外，它表示真正的 0 与无法显示的小数值 0.0000 不同）。

为了用同一显示格式，当矩阵中的最大元素小于 0.001 或其最小元素大于 1000 时，MATLAB 会把其中小于 0.001 或大于 1000 的公因子提出来，如键入由两个很小的数组成的矩阵

```
f=[0.000073,   5.33e-6]
```

得　f = 1.0e-004 *

```
[0.73     0.0533]
```

如果矩阵中出现大小差别很多的元素，则显示时将以大元素优先，小元素就只能显示很少的有效位，甚至成为 0.0000，这时不要误以为它是 0。可以用显示单个元素的命令来得到它的准确值，也可改用长格式（format long）来显示整个矩阵。

2.2　矩阵的初等运算

2.2.1　矩阵的加减乘法

矩阵算术运算的书写格式与普通算术相同，包括加、减、乘、除，也可用括号来规定运算的优先次序。但它的乘法定义与普通数（标量）不同。相应地，作为乘法逆运算的除法也不同，有左除（\）和右除（/）两种符号。

两矩阵的相加（减）就是其对应元素的相加（减），因此，要求相加的两矩阵的阶数必须相同。检查矩阵阶数的 MATLAB 语句是 size，例如：

键入　[n, m]=size(fb2)

得　n=6　　m=5　　（6 行 5 列）

如果要自己编写矩阵 ***A*** 和 ***B*** 相加（减）的程序，就必须先求 ***nA***，***mA***，***nB***，***mB***，并检验是否满足 ***nA=nB*** 和 ***mA=mB***。确认无误后再按对应元素相加（减），得出 ***C=A+B***（或 ***A-B***）。如果阶数检验不合格，则显示出错。当两个相加矩阵中有一个是标量时，MATLAB 承认算式有效，它自动把该标量扩展成同阶等元素矩阵，与另一矩阵相加。例如：

键入　X=[-1 0 1]; Y=X-1

得　Y = -2　-1　0

如果已经知道 X 是一维矩阵（数组），也可以用

```
l=length(X)
```

来求它的长度。注意，size 有 2 个输出量，而 length 只有 1 个输出量。length 不区分列或行，所以，作加减法阶数检验时只能用 size。

现在来看矩阵的乘法。$n\times p$ 阶矩阵 A 与 $p\times m$ 阶矩阵 B 的乘积 C 是一个 $n\times m$ 阶矩阵，它的任何一个元素 $C(i, j)$的值为 A 阵的第 i 行和 B 阵的第 j 列对应元素乘积的和。即

$$C(i, j)=A(i, 1)B(1, j)+A(i, 2)B(2, j)+\cdots+A(i, p)B(p, j)=\sum_k A(i, k)B(k, j)$$

式中的乘号是普通数（标量）的乘号。p 是 A 阵的列数，也是 B 阵的行数，也称为两个相乘矩阵的内阶数，两矩阵相乘的必要条件是它们的内阶数相等。不难看出，对于标量 A，B，因为 n，p，m 均为 1，矩阵乘法就退化为普通数的乘法。

如果要自己编写矩阵 A 和 B 相乘的程序，就必须先求 nA,mA,nB,mB,并检验 mA 是否等于 nB，确认无误后再按上式把对应元素相乘后累加，得出 $C(i, j)$。分别取 i 从 1 到 nA，j 从 1 到 mB，得出 $nA\times mB$ 个 C 元素，排成矩阵形式，得到 C。

实际上，MATLAB 已将上述矩阵加、减、乘的程序编程为内部函数，只要用+、-、* 做运算符号就包含了检查阶数和执行运算的全过程，而且，其运算对复数有效。

由此可见，前面定义的 X 和 Y 是不能相乘的，因为它们的内阶数分别为 3 和 1。如果两个乘数之一是标量，则 MATLAB 不检查其内阶数，而用该标量乘以矩阵的每个元素。如：

键入　　pi*X

得　　ans = -3.1416　　0　　3.1416

若把 Y 转置，成为 3×1 阶，则内阶数与 X 的相同，就可求

```
X*Y'
```

得　　ans =　2

不难用心算来检验其正确性。这个式子可读成 X 左乘 Y'。现在让 X 右乘 Y'，这时两者的内阶数都是 1，而外阶数（定义中的 nA 和 mB）都成了 3。于是有

```
Y'*X
```

得

```
ans =   2   0   -2
        1   0   -1
        0   0    0
```

显然，X 左乘和右乘 Y'所得的结果是完全不同的。只有单位矩阵例外，单位矩阵乘以任何矩阵 A（其阶数为 $nAmA$）时，不管是左乘还是右乘，其积仍等于该矩阵。即

```
eye(nA)*A=A
A*eye(mA)=A
```

读者可按单位矩阵的定义自行检验其正确性。

设有三个线性方程组成的联立方程组：

$$x_1+2x_2+3x_3=2$$
$$3x_1-5x_2+4x_3=0$$
$$7x_1+8x_2+9x_3=2$$

令　a=[1, 2, 3; 3, -5, 4; 7, 8, 9];　X=[x_1, x_2, x_3];　b=[2; 0; 2]

则这个联立方程组就可表为简洁的矩阵形式

$$aX'=b$$

给出 a 和 X，可以用乘法求出 b。但若给出 a 和 b 求 X，那就需要逆运算（矩阵除法）了。

2.2.2　矩阵除法及线性方程组的解

在线性代数中，没有除法，只有逆矩阵。矩阵除法是 MATLAB 从逆矩阵的概念引申来的。先介绍逆矩阵的定义，对于任意 $n\times n$ 阶方阵 A，如果能找到一个同阶的方阵 V，使

$$AV=I$$

其中，I 为 n 阶的单位矩阵 eye(n)。则 V 就是 A 的逆阵。数学符号表示为

$$V=A^{-1}$$

逆阵 V 存在的条件是 A 的行列式 det(A)不等于 0，V 的最古典的求法为高斯消去法，可参阅线性代数书。MATLAB 已把它做成了内部函数 inv，键入

```
V=inv(A)
```

就可得到 A 的逆矩阵 V。如果 det(A)等于或很接近于零，MATLAB 会显示出出错或警告信息：

"A 矩阵病态（ill-conditioned），结果精度不可靠"。

现在来看方程 D*X=B，设 X 为未知矩阵，在等式两端同时左乘以 inv(D)，即

inv(D)*D*X = inv(D)*B

等式左端 inv(D)*D=I，而 I*X=X，因此上式成为

X = inv(D)*B = D\B

把 D 的逆阵左乘以 B，MATLAB 就记作 D\，称之为"左除"。从 D*X=B 的阶数检验可知，B 与 D 的行数相等，因此，左除时的阶数检验条件是：两矩阵的行数必须相等。

如果原始方程的未知矩阵在左而系数矩阵在右，即

X*D = B

则按上述同样的方法可以写出

X = B*inv(D) = B/D

把 D 的逆阵右乘以 B，记作/D，称之为"右除"。同理，右除时的阶数检验条件是：两矩阵的列数必须相等。

下面来看矩阵左右乘除的一些示例。设 A=[1，2，3；4，5，6]，B=[2，4，0；1，3，5]，D=[1，4，7；8，5，2；3，6，0]

其乘除的结果列于表 2.2 中。

表 2.2　矩阵乘除法的示例

算　式	答　案
A*B	??? Error using ==> * Inner matrix dimensions must agree.（内阶数必须相等）
A'*B	6　16　20 9　23　25 12　30　30
A*B'	10　22 28　49

续表

算式	答案
D\A	??? Error using ==> \ Matrix dimensions must agree.（行数不等）
D\A'	-0.0370 0 0.5185 1.0000 -0.1481 0.0000
A/D	0.4074 0.0741 0.0000 0.7407 0.4074 0.0000

矩阵除法可以用来方便地解线性方程组。例如要求下列方程组的解 $x=[x_1; x_2; x_3]$。

$$6x_1+3x_2+4x_3=3$$
$$-2x_1+5x_2+7x_3=-4$$
$$8x_1-4x_2-3x_3=-7$$

此式可写成矩阵形式 $Ax=B$，求解的 MATLAB 程序为

```
A = [6, 3, 4; -2, 5, 7; 8, -4, -3]; B = [3; -4; -7]; x = A\B
```

得

```
x =  0.6000
     7.0000
    -5.4000
```

MATLAB 中的除法还可以用来解方程数不等于未知数个数的情况。比如再加上一个方程

$$x_1+5x_2-7x_3=9$$

这时系数矩阵 A1 的阶数为 4×3，不难看出 A1 的行数 nA1 是方程数，其列数 mA1 是未知数的个数，nA1>mA1，说明方程组是超定的，方程无解。照样列 MATLAB 程序

```
A1 = [6, 3, 4; -2, 5, 7; 8, -4, -3; 1, 5, -7]; B1 = [3; -4; -7; 9]; x1 = A1\B1
```

答案为

```
x1 = -0.1564
      1.0095
     -0.6952
```

它并未显示出错信息，却给出了解，这怎么可能呢？实际上，这时 MATLAB 给出的是最小二乘解。把这个 x1 代入方程组，肯定任何一个方程都不满足，都可得出 1 个误差，把这 4 个误差的平方相加开方，称为均方差。解 x1 保证比其他任何解所得的均方差都小。

MATLAB 中的除法还可以用来解方程数少于未知数个数的情况，A1 矩阵的 nA1<mA1，说明方程组是不定的，它有无穷个解。此时，仍然可用除法符号来求出解。这个解是满足方程的，但它不是惟一解。它是令 x1 中某个或某些元素为 0 的一个特殊解。

2.2.3 矩阵的乘方和幂次函数

MATLAB 的运算符 *、/、\、和^，指数函数 expm、对数函数 logm 和开方函数 sqrtm 是对矩阵进行的，即把矩阵作为一个整体来运算。除此之外，其他 MATLAB 函数都是对矩阵中的元素分别进行，英文直译为数组运算(Array Operations)，较准确的意义应为“元素群运算”，将在 2.3 节讨论。

在幂次运算时矩阵可以作为底数，指数是标量。这是矩阵乘法的扩展，为了保证做乘

法时内阶数相同，作为底数的矩阵必须是方阵。矩阵也可以作为指数，底数是标量，此时矩阵仍然应是方阵。底数和指数不能同时为矩阵，如果这样输入，将显示出错信息。表 2.3 给出了一些语句及其结果。注意 sqrtm 与 sqrt、expm 与 exp、logm 与 log 的不同，也即矩阵整体运算和矩阵元素群运算的不同。

表 2.3　矩阵整体运算的例及其结果

键入语句	输出结果	说　明
D^2	54　66　15 54　69　66 51　42　33	按矩阵运算
2.^D	2　16　128 256　32　4 8　64　1	按元素群运算
D^s	??? Error using ==> ^ Matrix dimensions must agree	非法运算
u1=sqrtm(s)	u1= 0.5537 + 0.4644i　0.8070 - 0.2124i 1.2104 - 0.3186i　1.7641 + 0.1458i	按矩阵运算 可用 u1*u1=s 检验
u2=sqrt(s)	u2=　1.0000　1.4142 1.7321　2.0000	按元素群运算， u2*u2≠s u2.*u2=s(见 2.3 节)
v1=expm(s)	v1=　51.9690　74.7366 112.1048　164.0738	按矩阵运算 可用 logm(v1)=s 检验
v2=exp(s)	v2=　2.7183　7.3891 20.0855　54.5982	按元素群运算 可用 log(v1)=s 检验
logm(D)	1.2447　-0.9170　2.8255 1.6044　2.5760　-1.9132 -0.7539　1.1372　1.6724	按矩阵运算
log(D)	0　1.3863　1.9459 2.0794　1.6094　0.6931 1.0986　1.7918　-Inf	按元素群运算

注：此表中 $s=\begin{bmatrix}1&2\\3&4\end{bmatrix}$，$D=\begin{bmatrix}1&4&7\\8&5&2\\3&6&0\end{bmatrix}$

2.2.4　矩阵结构形式的提取与变换

在做矩阵运算时，往往需要提取其中的某些特殊结构的元素，来组成新的矩阵；有时则要改变矩阵的排列。除了在 2.1 节讲过的提取行、列和将行列转置的语句之外，MATLAB 还提供了一些改变矩阵结构的函数。这些函数列于表 2.4 中，同时给出了相应的例。

表 2.4 矩阵结构形式提取和变换语句

函数名	功能说明	语句	结果
fliplr	矩阵左右翻转	B=fliplr(A)	B=0 6 1 8 1 7 5 3 2 2 9 4
flipud	矩阵上下翻转	B=flipud(A)	B=4 9 2 2 3 5 7 1 8 1 6 0
reshape	阶数重组(元素总数不变)	B=reshape(A，2，6)	B= 8 4 5 6 2 1 3 1 9 7 0 2
rot90	矩阵整体反时针旋转 90°	B=bot90(A)	B= 0 1 2 6 7 2 1 5 9 8 3 4
diag	提取或建立对角阵	B=diag(A)	B= [8，5，2]
tril	取矩阵的左下三角部分	B=tril(A)	B =8 0 0 0 3 5 0 0 4 9 2 0
triu	取矩阵的右上三角部分	B=triu(A)	B= 8 1 6 0 0 5 7 1 0 0 2 2
:	将元素按列取出排成一列	B=A(:)'	B=8 3 4 1 5 9 6 7 2 0 1 2

注：表中 $A=\begin{bmatrix}8&1&6&0\\3&5&7&1\\4&9&2&2\end{bmatrix}$

2.3 元素群运算

元素群运算能大大简化编程，提高运算的效率，是 MATLAB 优于其他许多语言的一个特色。

2.3.1 数组及其赋值

数组通常是指单行或单列的矩阵，一个 N 阶数组就是 $1\times N$ 阶或 $N\times 1$ 阶矩阵。一个 N 阶数组可以表述一个 N 维向量。因为一个三维空间的向量可用它在三个坐标轴上的投影来表示，即 $v=[vx, vy, vz]$，也即表示为三阶的数组。这个概念也可扩展到 N 维空间的向量。

在求某些函数值或曲线时，常常要设定自变量的一系列值，例如，设时间 t 从 0 到 1 之间，每隔 0.02 秒取一个点，共 51 个点，是 1×51 阶的数组。如果逐点给它赋值，将非常麻烦。MATLAB 提供了两种为等间隔数组赋值的简易方法。

1．用两个冒号组成等增量语句，其格式为：t=[初值：增量：终值]。如：

键入 t=[0: 0. 02: 1]

得　　t = 0　0.020　0.040　…　…　0.960　0.980　1.000

此语句中的增量也可设为负值，此时初值要比终值大。如：

键入　　z=10:-3:-5

得　　z = 10　7　4　1　-2　-5

当增量为 1 时，这个增量值可以略去，因而该语句只有一个冒号。如

键入　　k=1:6

得　　k = 1　2　3　4　5　6

2. 用 linspace 函数。其格式为

➢ linspace（初值、终值、点数）

键入　　theta = linspace(0, 2*pi, 9)

得　　theta = 0　0.7854　1.5708　2.3562　3.1416　3.9270　4.7124　5.4978　6.2832

在圆周上 0 和 2*pi 实际上是同一个点，所以，这个命令是把圆周分为 8 份。

有时要求自变量按等比级数赋值。在设频率轴时往往如此，这时可用 logspace 函数。例如：

键入　　w = logspace(0, 1, 11)

得　　w = 1.0000　1.2589　1.5849 … … 6.3096　7.9433　10.0000

它的意义是从 10 的 0 次幂到 1 次幂之间按幂等分（即数是等比的）为 11 个点。

2.3.2　元素群的四则运算和幂次运算

元素群运算也就是矩阵中所有元素按单个元素进行运算。为了与矩阵作为整体的运算符号相区别，要在运算符*、／、＼、^前加一点符号“.”，以表示在做元素群运算。参与元素群运算的两个矩阵必须是同阶的（只有标量除外，它会自动扩展为同阶矩阵参与运算），表 2.5 中表示两个 1×3 阶的元素群运算的结果。因为数取得很简单，可用心算加以检验。

表 2.5　简单的元素群运算

运　算　式	输 出 结 果	思　　考
Z=X.*Y	Z = 4　10　18	思考问题：X*Y 能成立吗
Z=X.\Y	Z =4.0000　2.5000　2.0000	元素群有没有左除右除之分
Z=X.^Y	Z = 1　32　729	思考问题：X^Y 能成立吗
Z=X.^2	Z = 1　4　9	思考问题：X^2 能成立吗
Z=2.^[X Y]	Z = 2　4　8　16　32　64	思考问题：2^[X Y]能成立吗

注：表中 X =[1, 2, 3]; Y=[4, 5, 6];

元素群的幂次运算也就是各个元素自行作幂次运算，对每个元素而言，这种运算和对标量运算一样，所以很容易判定它的正确性。

表 2.6 中参与元素群运算的是一个 3×3 阶的方阵，这是为了便于比较矩阵中的元素运算和矩阵整体运算的不同（因为非方阵是不能按整体作矩阵乘幂运算的）。从中也可以看出，不能将元素群运算称为数组运算，因为这里参加运算的是一个 3×3 阶矩阵而不是数组。

表 2.6 元素群和矩阵幂次运算的比较

输入算式	D	D^3	D.^3	3.^D
输出结果	1 4 7 8 5 2 3 6 0	627 636 510 804 957 516 486 612 441	1 64 343 512 125 8 27 216 0	3 81 2187 6561 243 9 27 729 1

注：表中 $D=\begin{bmatrix}1&4&7\\8&5&2\\3&6&0\end{bmatrix}$

特别注意 D^3 和 D.^3 结果的不同。3.^D 与 3^D 也完全不同，3^D 是合法运算，读者可自行实践并对其结果做出解释，不过这是比较难的题目，不作要求。

2.3.3 元素群的函数

前面已指出，大部分的 MATLAB 函数都适用于做元素群运算，只有专门说明的几个除外。那就是*、/、\、^ 运算符和 sqrtm、expm、logm 三个函数。表 2.7 基本函数库中的常用函数都可用于元素群运算，也即其自变量都可以是任意阶的矩阵。

表 2.7 基本函数库(elfun)（未标注输入变元的为单输入单输出函数）(c)

三角函数	sin	正弦	cos	余弦
	tan	正切	asin	反正弦
	acos	反余弦	atan	反正切
	atan2(x，y)	4 象限反正切	Sinh	双曲正弦
	cosh	双曲余弦	tanh	双曲正切
	acosh	反双曲余弦	atanh	双曲正切
	asinh	反双曲正弦	sec	正割
	csc	余割	cot	余切
	asec	反正割	acsc	余割
	acot	反余切	sech	双曲正割
	csch	双曲余割	coth	双曲正切
	asech	反双曲正割	acsch	反双曲余割
	acoth	反双曲正切		
指数函数	exp	以 e 为底的指数	log	自然对数
	log2	以 2 为底的指数	log10	以 10 为底的对数
	pow2	2 的幂	sqrt	方根
	nextpow2	比输入数大而最近的 2 的幂		
复数	abs	绝对值和复数模值	angle	相角
	real	实部	imag	虚部
	conj	共轭复数	isreal	是实数时为真
	unwrap	去掉相角突变	cplxpair	按复数共轭对排序元素群
取整函数	round	四舍五入为整数	fix	向 0 舍入为整数
	floor	向∞舍入为整数	ceil	向∞舍入为整数

续表

取整	sign	符号函数	rem(a，b)	a 整除 b，求余数
函数	mod(x，m)	x 整除 m 取正余数		

下面的例子可以说明利用元素群运算的优越性。例如，要求列出一个三角函数表。这在 MATLAB 中只要两个语句：

键入　`x=[0:0.1:pi/4]';[x, sin(x), cos(x), tan(x)]`

第一条语句把数组 x 赋值，经转置后成为一个列向量。因为 sin、cos、tan 函数都对元素群有效，得出的都是同阶的列向量。第二条语句把 4 个列向量组成一个矩阵，并进行显示。

得
```
0          0          1.0000     0
0.1000     0.0998     0.9950     0.1003
0.2000     0.1987     0.9801     0.2027
0.3000     0.2955     0.9553     0.3093
0.4000     0.3894     0.9211     0.4228
0.5000     0.4794     0.8776     0.5463
0.6000     0.5646     0.8253     0.6841
0.7000     0.6442     0.7648     0.8423
```

第一列是 x，以下各列依次是 sin(x)，cos(x)，tan(x)。如果要加一个表头，第二条语句可改成两条如下的显示语句：

```
disp('      x        sin(x)       cos(x)       tan(x)  ')
disp([x, sin(x), cos(x), tan(x)])
```

disp 后括号内引号中的内容是直接显示的，放入空格就显示空格，放入汉字就显示汉字。后一句括号中没有引号，是变量名组成的矩阵，它就显示该矩阵中各变量的值。

2.4　逻辑判断及流程控制

2.4.1　关系运算

所谓关系运算是指两个元素之间数值的比较，一共有表 2.8 所示的 6 种可能。

表 2.8　关系运算

<	<=	>	>=	==	~=
小于	小于等于	大于	大于等于	等于	不等于

关系运算的结果只有两种可能，即 0 或 1。0 表示该关系式为“假”，即它不成立；1 表示该关系式为“真”，即该关系式是正确的。例如，键入关系式：

```
a = 2+2= =4
```

得　`a = 1`

注意，前面的单个等号表示赋值，后面的双等号则表示关系运算。式中 2+2==4 是关系运算，它的优先级高，要先算，算出的结果给 a 赋值，为了改善可读性，最好加上括号，写成 a=(2+2= =4)，表明是把括号内的关系式的结果给 a 赋值。

MATLAB 中的关系运算都适用于矩阵，它是对矩阵的各个元素进行元素群运算，因此两个相比较的矩阵必须有相同的阶数，输出的结果也是同阶矩阵。例如

键入 `A=magic(6)`

得
```
A = 35   1    6    26   19   24
    3    32   7    21   23   25
    31   9    2    22   27   20
    8    28   33   17   10   15
    30   5    34   12   14   16
    4    36   29   13   18   1
```

要找到此矩阵中所有被 3 整除的元素，并在其位置上标以 1。可以用表 2.7 中的 rem 函数，rem(A，3)表示把 A 除以 3 的余数，余数为零就是整除。

键入 `p=(rem(A, 3)= =0)`

得
```
p = 0  0  1  0  0  1
    1  0  0  1  0  0
    0  1  0  0  1  0
    0  0  1  0  0  1
    1  0  0  1  0  0
    0  1  0  0  1  0
```

关系运算中还包括某些条件判断，例如判断矩阵元素中有无 NaN、Inf 值，矩阵是否实数阵、稀疏阵或空阵等，它们不能直接用上述 6 种关系符简单地表述，MATLAB 把它们编成了专用的函数以备直接调用，见表 2.9。

表 2.9 运算符和特殊字符库(ops) (n)

	符 号	意 义	符 号	意 义	符 号	意 义
数学及逻辑运算符	+	加	-	减	*	矩阵乘
	\	矩阵左除	/	矩阵右除	^	矩阵乘幂
	.*	矩阵元素乘	./	矩阵元素除	.^	矩阵元素乘幂
	( ) {}	优先，下标 输入参量	[ ]	矩阵，向量 输出变量	:	整行（列） 等增量赋值
	.	小数点	..	母目录	...	行命令延续符
	,	语句分割符，显示	;	语句分割符，不显示	=	赋值符
	'	转置，引用	!	操作系统命令	%	注释符
	==	关系相等符	< >	关系大小符	~=	关系不等符
	&	逻辑与	|	逻辑或	~	逻辑非
	xor	异或	kron	Kronecker 积		

逻辑字符检查	exist	检查变量或函数是否有定义	any	检查向量中有无非零元素
	all	检查向量中元素是否全为非零	find	找到非零元素的序号
	isnan	元素为 NaN 时得 1	isinf	元素为 Inf 时得 1
	isfinite	元素为有限值时得 1	isempty	矩阵为空阵时得 1
	isreal	矩阵为实数阵时得 1	issparse	矩阵为稀疏阵时得 1
	isstr	为文本字符串时得 1	isglobal	变量为全局变量时得 1

续表

位运算	bitand *	按位求“与”	bitcmp *	按位求“非”（补）
	bitor *	按位求“或”	bitmax *	最大浮点整数
	bitxor *	按位求“异或”	bitset *	设置位
	bitget *	获取位	bitshift *	按位移动
集合运算	union *	集合“合”	unique *	去除集合中的重复元素
	intersect*	集合“交”	setdiff *	集合“差”
	setxor *	集合“异或”	ismember *	是集合中的元素时为真

➢ [j,k]=find(p) 给出 p 矩阵中不为零的元素的两个下标，左端没有或只有一个变量，即 find(p)或 lp=find(p)给出 p 矩阵中不为零的元素的序号。矩阵元素是按列排序号的，先第 1 列，再接第 2 列……依次排完后，再确定它们的顺序号。一个 6×6 阶的矩阵的 36 个元素的序号排列如表 2.10 所示。因此，一个 n×m 阵中下标为(j, k)的元素，其序号为 l=(k-1)*n+j。

键入 `lp=find(p)'`

得 `lp = 2 5 9 12 13 16 20 23 27 30 31 34`

表 2.10 矩阵元素的序号排法

1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35
6	12	18	24	30	36

可以看出这些序号确实对应于 p 中的 1 元素。矩阵的序号（index）与下标（subscript）是一一对应的，其变换关系可由表 2.1 中的 ind2sub（读作 index to subscirpt）和 sub2ind 函数求得。

2.4.2 逻辑运算

逻辑量只能取 0（假）和 1（真）两个值。逻辑量的基本运算为与（&）、或（|）和非（~）三种。有时也包括异或（xor），不过异或可以用三种基本运算组合而成。两个逻辑量经此逻辑运算后的输出仍然是逻辑量，表示逻辑量的输入输出关系的表称为真值表，见表 2.11。

表 2.11 基本逻辑运算的真值表

运 算	A = 0		A = 1	
	B = 0	B = 1	B = 0	B = 1
A & B	0	0	0	1
A | B	0	1	1	1
~A	1	1	0	0
xor(A, B)	0	1	1	0

所有的算法语言中都有逻辑运算。MATLAB 的特点是将逻辑运算用于元素群，得出同阶的 0-1 矩阵。为了按列、按行判断一群元素的逻辑值，它又增加了两种对元素群的逻辑运

算函数，即 all（全为真）和 any（不全为假）。

现在来看逻辑式 u=p|~p，这是把 p 和“非”p 求“或”。~p 就是把 p 中的 0 元素换成 1，1 元素换成 0。在每个元素位置上，必有一个是 1，把 p 和~p“或”起来，一定是全 1 矩阵。

得

```
u = 1    1    1    1    1    1
    1    1    1    1    1    1
    1    1    1    1    1    1
    1    1    1    1    1    1
    1    1    1    1    1    1
    1    1    1    1    1    1
```

all 和 any 后的输入变量应为矩阵，它是按列运算的。从它们的定义可知

all(p) = 0	0	0	0	0	0	（列中有一个元素为 0 即得 0）
all(u) = 1	1	1	1	1	1	（列中元素为全 1 才得 1）
any(p) = 1	1	1	1	1	1	（列中有一个元素为 1 即得 1）

2.4.3 流程控制语句

计算机程序通常都是从前到后逐条执行的。但有时也会根据实际情况，中途改变执行的次序，称为流程控制。MATLAB 4.x 设有 3 种流程控制的语言结构。即 If 语句、While 语句和 For 语句。在 MATLAB 5.x 中是 4 种，增加了 Switch-case 语句。

1. If 语句

根据复杂程度，If 语句有 3 种形式：

● if（表达式）语句组 A，end

其流程见图 2.1(a)。执行到此语句时，计算机先检验 if 后的逻辑表达式，如为 1，它就执行语句组 A；如为 0，就跳过语句组 A，直接执行 end 后的后续语句。注意，这个 end 是决不可少的，没有它，在表达式为 0 时，就找不到继续执行的程序入口。

● if（表达式 1）语句组 A，else 语句组 B，end

其流程见图 2.1(b)。执行到此语句时，计算机先检验 if 后的（逻辑）表达式，如为 1，它就执行语句组 A；如为 0，就执行语句组 B。else 用来标志语句组 B 的执行条件，同时也标志语句组 A 的结束（免去了 end）。同样，最后的 end 是不可少的；没有它，执行完语句组 A 后，会找不到进入后续程序的入口。

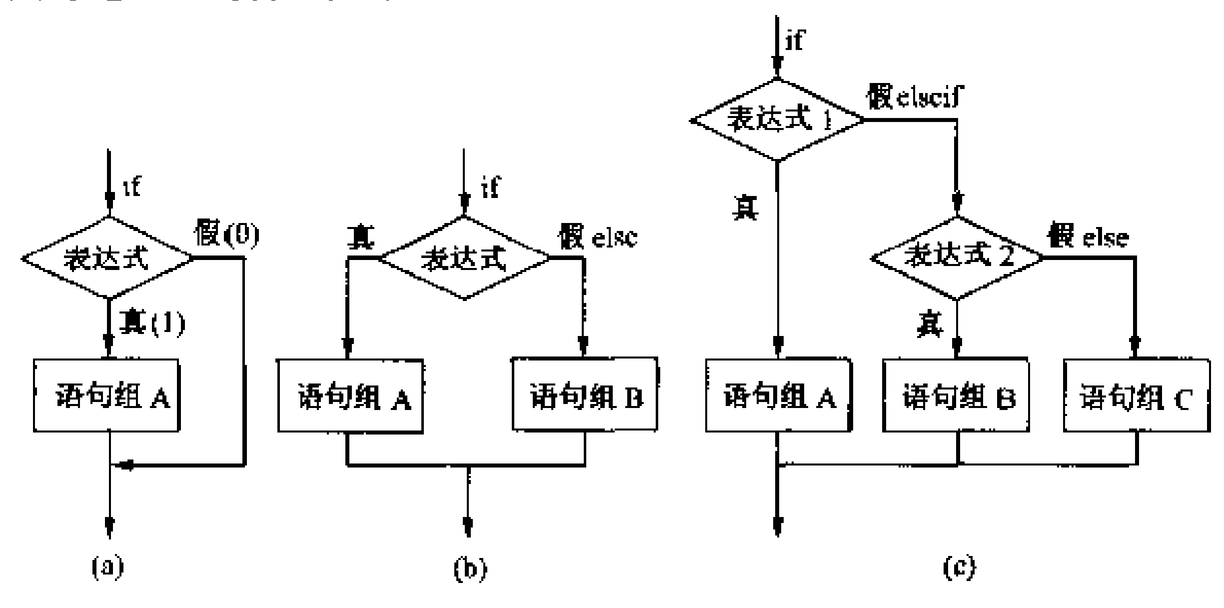


图 2.1 if 语句的 3 种程序结构形式

● if（表达式 1）语句组 A，elseif （表达式 2）语句组 B， else 语句组 C，end

其流程见图 2.1(c)。前两种形式的 if 语句都是两分支的程序结构，要实现两个以上分支的结构就得采用含 elseif 的结构。这里表示的是 3 分支的情况。在中间可加入多个 elseif 以形成多个分支。只是程序结构会显得冗长，MATLAB5.x 中的 Switch 语句可以用较简洁对称的形式实现多分支结构。

【例 2.1】 输入数 n，判断其奇偶性。

程序如下

```
n=input('n='), if  rem(n, 2)==0 A='even', else  A='odd', end
```

运行此程序时，程序要求用户输入一个数，然后它判断该数是奇数还是偶数。所以它共有两个出口。实际上这个程序并不全面，如果用户根本未键入任何数就回车，程序会判断为 odd。请读者考虑其原因。为了使程序在用户无输入时自动中止，可以把程序改为：

```
if  isempty(n)==1 A='empty', elseif rem(n, 2)==0 A='even', else  A='odd', end
```

实际上，这个程序仍不全面，它不能用于负数，请读者分析其原因。

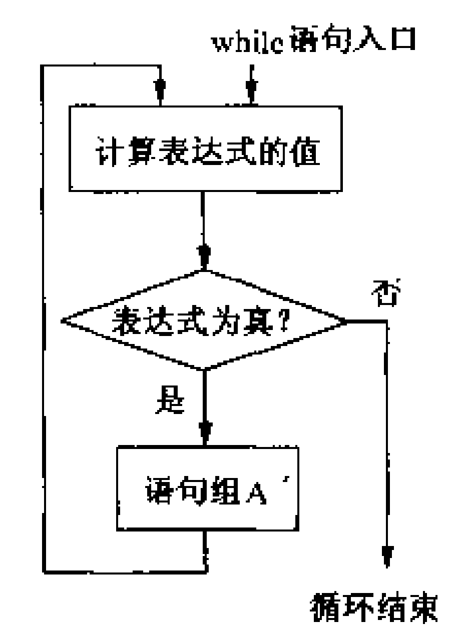


图 2.2　while 语句流程图

2. While 语句

while 语句的结构形式为：

➢ while （表达式）语句组 A，end

其流程见图 2.2。执行到此语句时，计算机先检验 while 后的逻辑表达式，如为 1，它就执行语句组 A；到 end 处后，它跳回到 while 的入口，再检验表达式；如还是 1，再执行语句组 A；周而复始，直到表达式不成立（结果为零）为止。此时就跳过语句组 A，直接执行 end 后的后续语句。与 if 语句的不同之处是它在分支中是循环地执行某个语句组，故称为循环语句。

【例 2.2】 求 MATLAB 中的最大实数。

解：设定一个数 x，让它不断增大，直到 MATLAB 无法表示它的值，只能表为 inf 为止。于是，可列出下列程序

```
x=1; while x~=inf,  x1=x; x=2*x; end,  x1
```

其中，先设 x=1，进入 while 循环，只要 x 不等于 inf，就把 x 加倍，直到 x=inf。如果把此时的 x 显示出来，它是无穷大，不是题中要找的数。要找的是变为无穷大之前的最大数。因此，在对 x 加倍之前，把它存在 x1 中，显示的 x1 就是要求的最大数。运行这行程序得

```
x1 =  8.9885e+307
```

系统的最大浮点实数为 $(2-\varepsilon)*2^{1023}$（见表 2.1），其十进制形式为

```
realmax = 1.7977e+308
```

两者数量级接近，但还是相差将近一倍，这是因为每次都把 x 翻一番，故求得的数可能比最大数小不到一半。如果把程序中的 x=2*x 改为 x=1.1*x，结果就会准确一些，得到

```
x1 = 1.783718732622142e+308
```

【例 2.3】 求 MATLAB 相对精度。

解：解的思路是让 y 不断减小，直至 MATLAB 分不出 1+y 与 1 的差别为止。其程序为

```
y=1;while 1+y>1, y1=y;y=y/2;end, y1
```

结果为

```
y1 =  2.220446049250313e-016
```

与 MATLAB 内部给出的浮点数相对精度 2^{-52}（见表 2.1）的十进制数相同。

3. for 语句

for 语句的结构形式为：

➢ for k= 初值：增量：终值　语句组 A，end

即它把语句组 A 反复执行 N 次。在每次执行时程序中的 k 值不同。

N ＝ 1+（终值-初值）/增量

【例 2.4】 用 for 语句求三角函数表。

程序如下：

```
for x=0:0.1:pi/4 disp([x, sin(x), cos(x), tan(x)]), end
```

所得的结果将和前面的答案相同。这也可以看出，MATLAB 的元素群运算功能与一个 for 循环相当。由于它不需每次检验表达式，运算速度比 for 语句快得多。但是不能认为它可全部取代 for 语句，由下例可以看出。

【例 2.5】 列出构成 Hilbert 矩阵的程序。

完成这个程序需要两重循环：

```
n=input('n='), format rat
for i=1:n, for j=1:n, h(i, j)=1/(i+j-1);end, end, h
```

执行时，先按提示输入 n，比如输入 5。

结果为

```
h =   1        1/2       1/3       1/4       1/5
      1/2      1/3       1/4       1/5       1/6
      1/3      1/4       1/5       1/6       1/7
      1/4      1/5       1/6       1/7       1/8
      1/5      1/6       1/7       1/8       1/9
```

为了改善可读性，对于流程控制语句，最好用缩进的方法写程序。本例中应写成：

```
format rat, n=input('n='),
for i=1:n
    for j=1:n
        h(i, j)=1/(i+j-1);
    end
end
h
```

由于现在是在 MATLAB 命令窗中直接输入程序，因此，不得不把它写在一行中。此时要注意，在 if、for、while 与表达式之间应留空格，在表达式与语句组之间必须用空格或逗号分隔，而在语句组的后面，必须要用逗号或分号来与 end 或 else 相分隔。否则，MATLAB 会显示出错信息并中止运行。

break 是中止循环的命令，在循环语句中，可用它在一定条件下跳出循环，它是常常用

到的。在多重循环中，break 只能使程序跳出包含它的最内部的那个循环。

4. Switch 语句

Switch-case-otherwise 语句是 MATLAB 5.x 中新扩展的。它是一种均衡实现的多分支语句，其基本语言结构可表为：

```
switch  表达式（标量或字符串）
case  值 1
语句组 A
case  值 2
语句组 B
………
otherwise
语句组 N
    end
```

当表达式的值（或字符串）与某 case 语句中的值（或字符串）相同时，它就执行该 case 语句后的语句组，然后直接跳到终点的 end。case 语句可以有 N-1 个，如果没有任何一个 case 值能与表达式值相符，则将执行 otherwise 后面的语句组 N。

例如，判断输入数 n 的奇、偶、空的程序可用 Switch 语句写成如下：

```
switch mod(n, 2), case 1, A='奇', case 0, A='偶', otherwise, A='空', end
```

注意，把它写成单行命令时的标点格式，其中有些逗号可以用分号代替，但不得省略。另外，为了包含负数中的奇数，将前面例中的 rem 改为 mod，读者可从 rem（-3，2）和 mod（-3，2）的差别得知这样做的原因。在正式写程序时，case 语句必须写在行首，以增强程序的可读性。

2.5　基本绘图方法

MATLAB 可以根据给出的数据，用绘图命令在屏幕上画出其图形，通过图形对科学计算进行描述。这是 MATLAB 独有的优于其他语言的特色。它可选择多种类型的绘图坐标，可以对图形加标号、加标题、或画上网状标线。这些命令属于 graph2d 函数库，另外，还有一些命令可用于屏幕控制，坐标比例选取以及在打印机上进行硬拷贝等等。这些命令放在 graphics 函数库中。三维及颜色绘图命令放在 graph3d 函数库中。还有一些特殊绘图命令放在 specgraph 函数库中。本书不可能介绍所有的命令，但大部分命令会在本书中涉及，下面分别进行讨论。

2.5.1　直角坐标中的两维曲线

plot 命令用来绘制 *X-Y* 坐标中的曲线。它是一个功能很强的命令。输入变量不同，可以产生很多不同的结果。

1. plot(y)——输入一个数组的情况

如果 y 是一个数组，函数 plot(y)给出线性直角坐标的二维图，以 *y* 中元素的下标作为 *X*

坐标，y 中元素的值作为 Y 坐标，一一对应画在 X-Y 坐标平面图上，而且将各点以直线相联。例如，要画出 10 个随机数的曲线。可列出：

```
y=5*(rand(1, 10)-.5)
y = -1.4052 -2.2648 0.8943 0.8965 2.1735 -0.5825 0.0971 1.6548 -2.3271 -2.2327
```

由 Rand 函数产生的随机数的最大值为 1，最小数为 0，平均值为 0.5。所以 y 的最大值为 2.5，最小值为-2.5，平均值为 0。键入 plot(y)，MATLAB 会产生一个图形窗，自动规定最合适的坐标比例绘图。X 方向是横坐标，从 1 到 10，Y 方向范围则是-4 到 4，并自动标出刻度。可以用 title 命令给图加上标题，用 xlabel、ylabel 命令给坐标轴加上说明，用 text 或 gtext 命令可在图上任何位置加标注，也可用 grid 命令在图上打上坐标网格线。

键入
```
title('my first plot')
xlabel('x'), ylabel('Y')
grid
```

这时形成如图 2.3 所示的图。

2. plot(x，y)——输入两个数组的情况

如果数组 x 和 y 具有相同长度，命令 plot（x，y）将绘出以 x 元素为横坐标，y 元素为纵坐标的曲线。例如，设 t 为时间数组 t=0：0.5：4*pi，y 是一个随 t 作衰减振荡的变量，y=exp(-0.1*t).*sin(t)，则 plot(t，y)就以 t 为横坐标，y 为纵坐标画曲线。如图 2.4 中的实线曲线。若设 y1=exp(-0.1*t).*sin(t+1)，则由 plot(t，y1，':')画出的曲线，其正弦波的相位超前了 1 弧度。因此，其波形如图 2.4 中的虚线曲线所示。实际上，在绘制第二条曲线时，如不加别的命令，第一条曲线就自动消失了。不会有两根曲线同在一张图中出现。为了在一张图中绘制多条曲线，要用后面所说的办法。

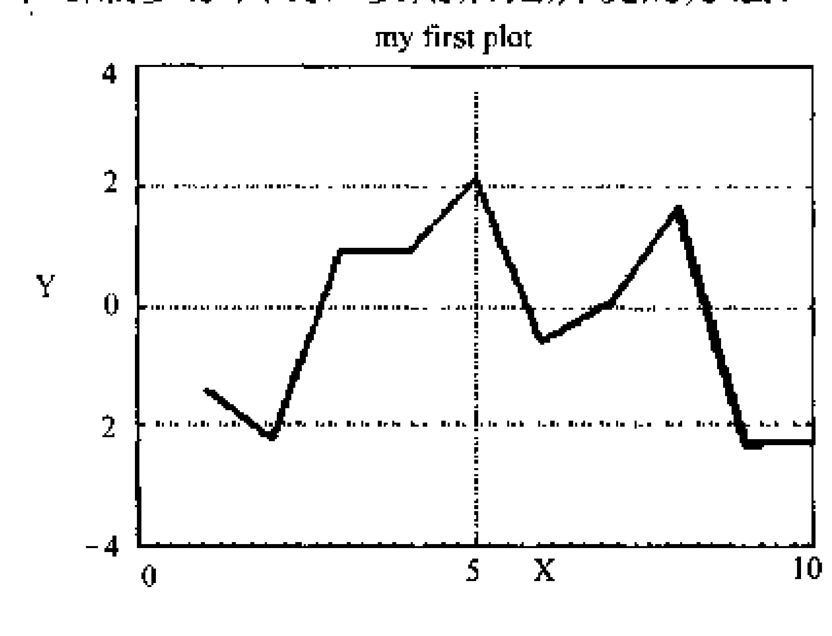


图 2.3 第一张简单的随机数图

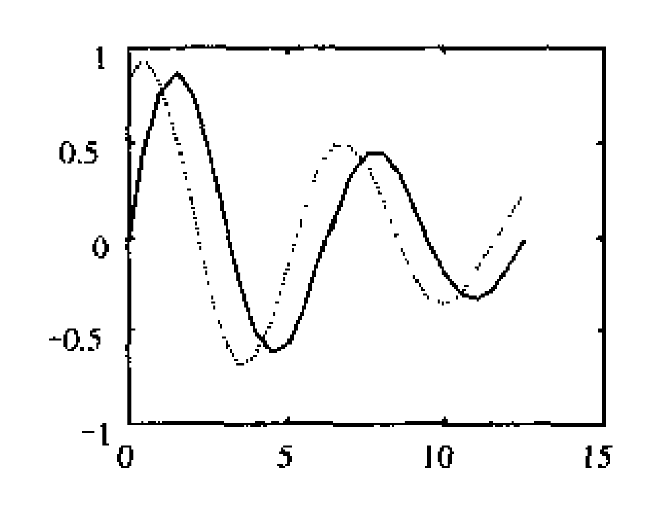


图 2.4 两根曲线画在同一图上

2.5.2 线型、点型和颜色

MATLAB 会自动设定所画曲线的颜色和线型。如果用户对线型的默认值不满意，可以用命令控制线型。也可以根据需要选取不同的数据点的标记。为了设定线型，在输入变量组的后面，加一个引号，在引号内部放入线型和颜色的标识符，如：

```
plot (x, y, '*b')
```

这样绘出的图线，其数据点处均用 * 作蓝色标记，而各点之间不再连以直线。

```
plot (x1, y1, ':y'), plot(x2, y2, '+r')
```

绘出的第一条曲线是黄色的点线，第二条曲线的数据点标记为红色的“+”号。其他线型、点型和颜色见表 2.12。

表 2.12　线型、点型和颜色

标　识　符	颜　　色	标　识　符	线型和点型
y	黄	.	点
m	品红	o	圆圈
c	青	x	x 号
r	红	+	+ 号
g	绿	-	实线
b	蓝	*	星号
w	白	:	虚线
k	黑	-.	点划线
		--	长划线

2.5.3　多条曲线的绘制

在一张图上画多根曲线有 4 种方法，其中第 4 种方法是 MATLAB 5.x 中新增加的。

1．用 plot(t，[y1，y2，…])命令

该语句中 t 是向量，y=[y1，y2，…] 是矩阵，若 t 是列（行）向量，则 y 的列（行）长与 t 长度相同。y 的行（列）数就是曲线的根数。例如：

键入　plot(t, [y; y1])

就得出图 2.4 中的曲线。它会自动给曲线以不同的颜色。这种方法要求所有的输出量有同样的长度和同样的自变量向量。另外，它不便于用户自行设定线型和颜色。

2．用 hold 命令

在画完前一张图后用 hold 命令保持住，再画下一条曲线。如：

键入　plot(t, y), hold on, plot(t, y1, 'g')

执行此命令时，图形窗产生第一幅图形，同时，命令屏幕显示 Current plot held，图形处于保持状态。再执行 plot(t，y1，'g')，就把第二幅图以绿色的曲线叠合在同一张图上。

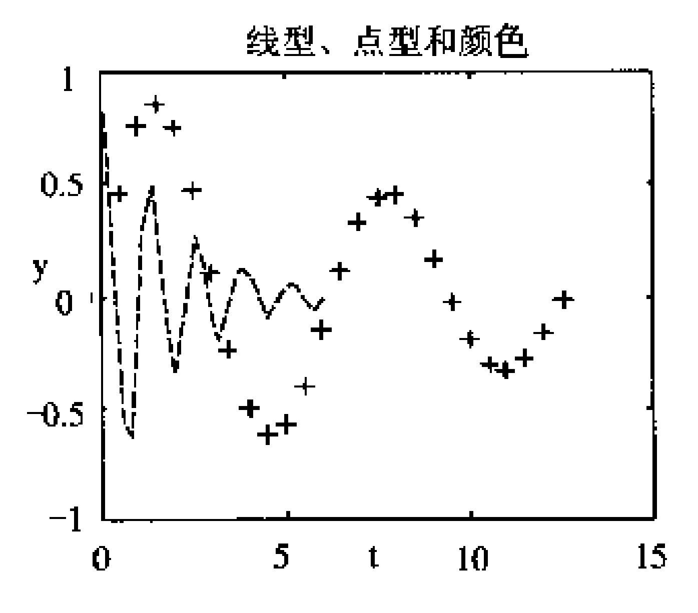


图 2.5　两组长度不同的[t，y]数据画在同一图上

用这种方法时两张图的变量长度可以各不相同。只要每张图自己的自变量和因变量同长即可。例如，再给一组数据[t2，y2]，其点数比[t，y]多，但占的时间却短。

键入

```
t2=0:.2:2*pi;y2=exp(-0.5*t2).*sin(5*t2+1); plot(t2,y2)
```

得出的图形为图 2.5 中较短的那条曲线（但线型不同）。用这种方法时，需注意两点：(1) 注意第一张图的坐标要适当，以保证能看清第二张图。因为用第一种方法

时，坐标系是系统自动按多根曲线的数据综合选取的，不会有选择不当的问题。(2) 注意及时解除保持状态，即键入 hold off；否则，以后的图都会叠加在此图上，造成混乱。

3．在 plot 后使用多输入变量

在 plot 后使用多输入变量所用的语句为：

➢ plot（x1，y1，x2，y2，…，x n，y n）

其中，x1，y1，x2，y2 等分别为数组对。每一对 X-Y 数组可以绘出一条图线，这样就可以在一张图上画出多条图线，每一组数组对的长度可以不同，在其后面都可加线型标志符。例如：

键入

```
plot (t, y, '+g', t2, y2, ': r')
title ('线型，点型和颜色')
xlabel ('时间'), ylabel ('Y')
```

执行这些语句就得到图 2.5。一根图线在数据点处用绿色的虚线做标记，另一根图线用红色的+号做标记。注意，这里用的是汉字标注，MATLAB 也照样把汉字标在图上。因为在引号中的内容，MATLAB 只作为一种代码来传递。

4．用 plotyy 命令

plotyy 是 MATLAB 5.x 中新增的函数，它设有两个纵坐标，以便绘制两个 y 尺度不同的变量，但 x 仍只用同一个比例尺，例如：

键入　　`y3=5*y2;  plotyy(t, y', t2, y3)`

就得到图 2.6。其中，左纵坐标是对 y 的，而右纵坐标是对 y3 的，纵坐标和曲线的标注可用 gtext 命令：

```
grid,  gtext ('t, t2')
gtext('y'), gtext ('y3')
```

gtext 命令用鼠标拖动来确定标注文字的位置，用起来比较方便。

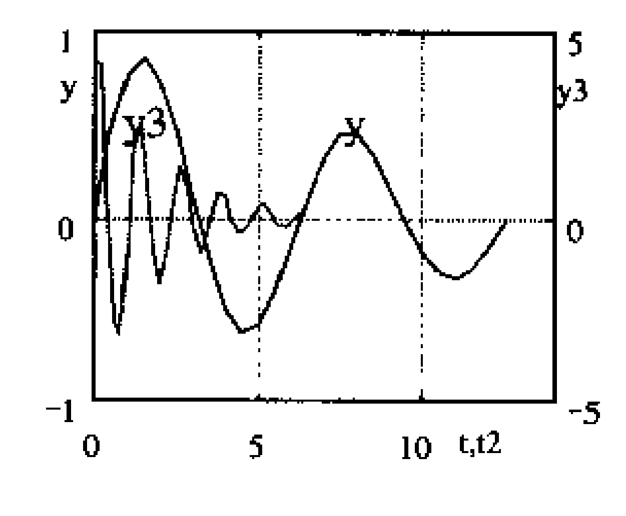


图 2.6　双纵坐标绘图

2.5.4　屏幕控制和其他二维绘图

1．图形屏幕控制命令（参看表 2.13）

图形屏幕可以开或关，可以开几处图形窗，也可以在一个图形窗内画出几幅分图，几幅分图也可用不同的坐标。以下几种命令可以实现图形窗口间的转换和清除。

• figure：打开图形窗口。MATLAB 中的第 1 幅图随 plot 命令自动打开，以后的 plot 命令都画在同一张图上。如要画在另一张新图上，就要用 figure 命令打开新的图形窗口。有了顺序为 1，2，3，…的几个图形窗后，再用 plot 语句，就要指明画在哪张图上，即键入 figure；表示打开第 i 幅图。否则，所有的图都会画在最后显示的那幅图上。

• clf：清除当前图形窗的内容（也可用 clg，但以后将被淘汰）

• hold：保持当前图形窗的内容，再键入 hold，就解除保持状态。这种拉线开关式的控

制有时会造成混乱，可以用 hold on 和 hold off 命令以得到确定的状态。

- close：关闭当前图形窗。
- close all：关闭所有图形窗。
- subplot（n，m，p）命令：将图形窗口分为 $n \times m$ 个子图，在第 p 个子图处绘制图形。

表 2.13　通用图形函数(graphics) (h)

图形窗的控制	figure	创建图形窗	shg	显示图形
	gcf	获取当前图形窗的句柄	refresh	刷新图形
	clf	清除当前图形窗	close	关闭图形窗
轴系的控制	axes	在任意位置创建坐标系	ishold	保持当前图形状态为真
	gca	获取当前坐标系的句柄	box*	形成轴系方向
	cla	清除当前坐标系		
图形对象	line	创建直线	surface	创建曲面
	patch	创建图形填充块	light *	创建照明
	image	创建图像		
图形句柄操作	set	设置对象特性	gcbo	获得回叫对象的句柄
	get	获得对象特性	gcbf	获得回叫图形的句柄
	reset	复位对象特性	drawnow	直接等待图形事件
	delete	删除对象	findobj *	寻找具有特定值的对象
	gco	获得当前对象的句柄	copyobj *	为图形对象及其子项作硬拷贝
工具	closereq	请求关闭图形窗	ishandle *	是图形句柄时为真
	newplot	说明 NextPlot 的 M 文件		
杂项	ginput	从鼠标作图形输入	uiputfile	给出存储文件的对话框
	graymon	设定图形窗灰度监视器	uigetfile	给出询问文件名的对话框
	rbbox	涂抹块	whitebg	设定图形窗背景色
	rotate	围绕指定方向旋转对象	zoom	二维图形的放大和缩小
	terminal	设定图形终端类型	warndlg	警告对话框

2．其他二维绘图命令（参看表 2.14）

在线性直角坐标系中绘出形式图的命令有 stem（绘脉冲图）、stairs（绘阶梯图）、bar（绘条形图）、errorbar（绘误差条形图）、hist（绘直方图）等。这些函数用法与 plot 相仿，但没有多输入变量形式。fill（t，y，'颜色标注符'）在曲线和坐标轴之间的封闭区填以指定的颜色。

下列程序把画面分成 4 个。

```
subplot(2, 2, 1), stem(t, y)
title('stem(t, y)'), pause
subplot(2, 2, 2), stairs(t, y)
title('stairs(t, y)'), pause,
subplot(2, 2, 3), bar(t, y)
title('bar(t, y)') , pause
```

```
subplot(2, 2, 4), fill(t, y, 'r')
title('fill(t, y, ''r'')')
```

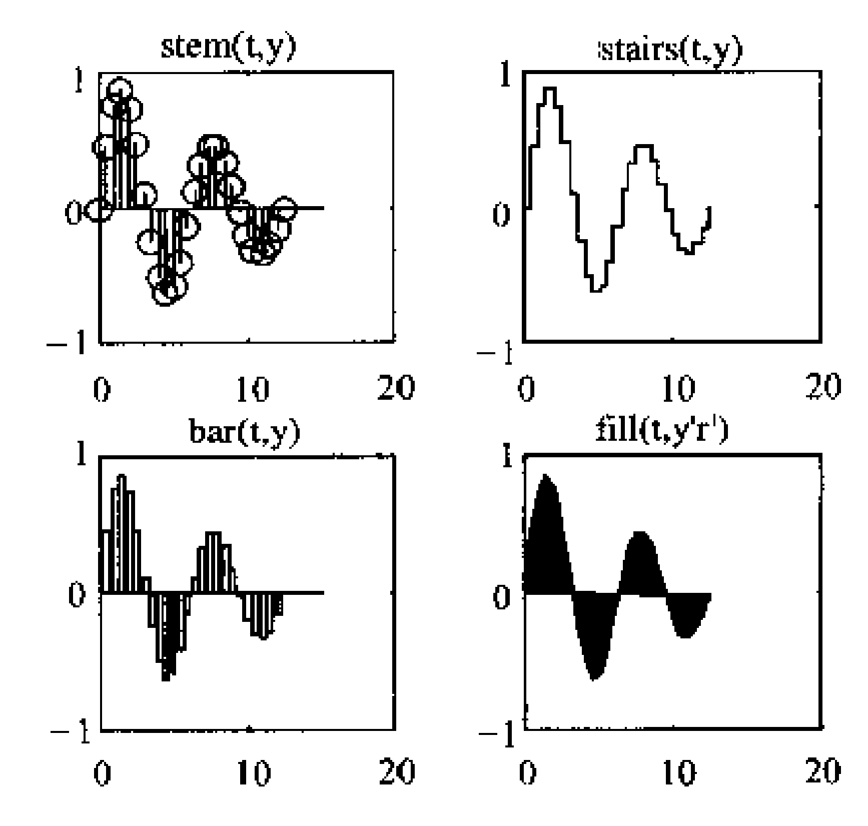


图 2.7 同一函数的几种不同的绘制形式

其运行的结果见图 2.7。读者不难从中弄清这几条绘图命令的意义。程序中最后一行，在 r 前后的引号写成两个引号，这是因为它是处在 title 后的引号内。MATLAB 规定，这种引号必须写成两个，以免混淆。

再键入 subplot(1，1，1)命令可取消子图，转回全屏幕绘图。

在对数直角坐标系中的绘图命令有 loglog，semilogx，semilogy 等，在极坐标系中的绘图命令有 polar。它们的用法与 plot 的基本用法相同。只是数据将画在不同类的坐标系上。可参看表 2.14。

表 2.14 二维图形函数库(graph2d) (p)

基本 X-Y 图形	plot	线性 X-Y 坐标绘图	polar	极坐标绘图
	loglog	双对数 X-Y 坐标绘图	plotyy	用左、右两种 Y 坐标画图
	semilogx	半对数 X 坐标绘图	semilogy	半对数 Y 坐标绘图
坐标控制	axis	控制坐标轴比例和外观	subplot	在平铺位置建立图形轴系
	hold	保持当前图形		
图形注释	title	标出图名（适用于三维图形）	gtext	用鼠标定位文字
	xlabel	X 轴标注（适用于三维图形）	legend	标注图例
	ylabel	Y 轴标注（适用于三维图形）	grid	图上加坐标网格（适用于三维）
	text	在图上标文字（适用于三维）		
打印	print	打印图形或把图存为 M 文件	orient	设定打印纸方向
	printopt	打印机默认选项		

• polar(theta，rho)为极坐标绘图，以角度 theta 为一个坐标，单位是弧度，另外一个是矢径 rho。在其后使用 grid 命令，可以绘出网状极坐标线。这个函数没有多输入变量形式。

• loglog 绘出以 log10-log10 为坐标刻度的对数图。

• semilogx 使用半对数刻度绘图，X 轴为 log10 刻度，Y 轴为线性刻度。

• semilogy 使用半对数刻度绘图，Y 轴为 log10 刻度，X 轴为线性刻度。

3. 虚数的绘图

当 plot（z）中的 z 为复数单变量时（即含有非零的虚部），MATLAB 把复数的实部作为X坐标，虚部作为Y坐标进行绘图，即相当于 plot（real（z），imag（z））。如果是双变量，如 plot（t，z)，则 z 中的虚数部分将被丢弃。要在复平面内绘出多条图线，必须用 hold 命令，或把多根曲线的实部和虚部明确地写出，作为 plot 函数的输入变元，即：

```
plot ( real ( z1 ), imag ( z1 ), real ( z2 ), imag ( z2 )).
```

例如要绘制 z=exp（(-1+i）*t）的复数图形，列出下面的程序：

```
figure(2)
```

```
z=exp((-0.1+i)*t);
subplot(2,2,1)
plot(z), pause
title('复数绘图 plot(z)')
subplot(2,2,2)
plot(t, z), pause
title('复数绘图 plot(t, z)')
subplot(2,2,3), polar(angle(z), abs(z))
title('polar(angle(z), abs(z))')
subplot(2,2,4), semilogx(t,z)
title('semilogx(t, z)')
```

得到图 2.8。其中，第一个子图画出了复数图形；第二个子图只画了 z 的实部随 t 的变化规律，第三个子图是用极坐标绘制的复数曲线；第四个子图说明了半对数坐标绘图的结果。

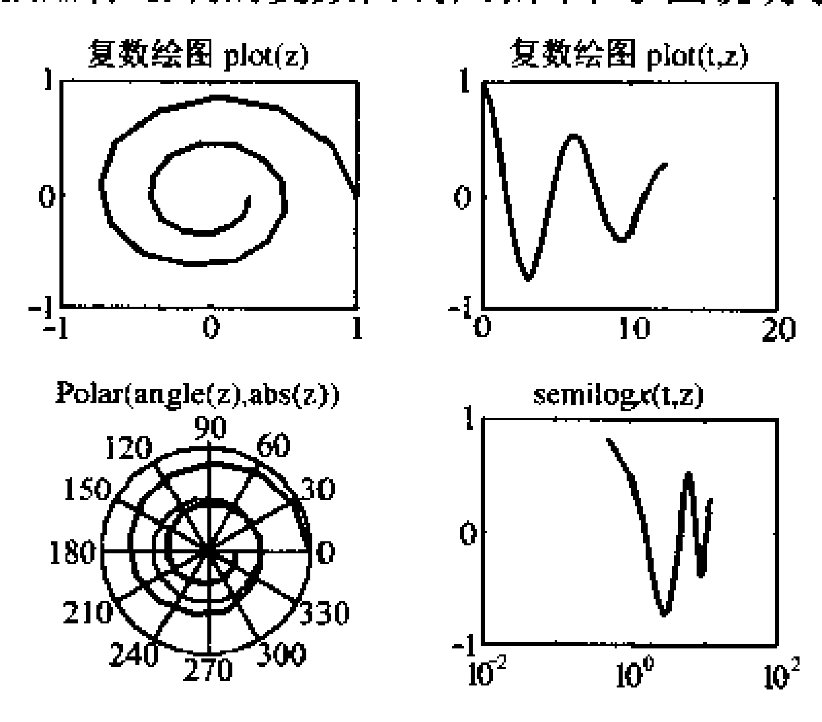


图 2.8　复数绘图及其他坐标轴绘图

4. 坐标比例和尺寸的设定——axis 命令

MATLAB 有根据输入数据自动设定坐标比例的功能。但在有些情况下，用户需要自行设定坐标比例并选择图形边界范围，这时可用 axis 命令。它有多种用法，由输入变量的不同而定。

V=axis，返回值为当前图形边界的 4 元行向量，即 V=[xmin，xmax，ymin，ymax]，如果当前图形是三维的，则返回值将是三维坐标边界的 6 元行向量。

axis(V)（其中 V 是一个 4 元向量），将坐标轴设定在 V 规定的范围内。

axis 的另外一个功能是控制图形的纵横比。axis('square')或 axis('equal')使屏幕上 x 与 y 的比例尺相同，在这种方式下，斜率 1 的直线的倾角为 45°，对于程序：

```
z=0:0.1:2*pi;x=sin(z);y=cos(z);
subplot(1, 2, 1), plot(x, y), subplot(1, 2, 2), plot(x, y), axis('equal')
```

虽然数据得出的是圆，但由于屏幕本身长宽不等，第一个子图得出的是椭圆。第二个子图由于函数 axis('equal')的作用，就成为圆（如图 2.9）。axis('normal') 将恢复正常的纵横尺寸比。

键入　　v=axis

可得　　v =　　-1　　1　　-1　　1

axis 命令的功能非常丰富，这里只介绍了一部分。要知道详情，可使用命令 help axis。

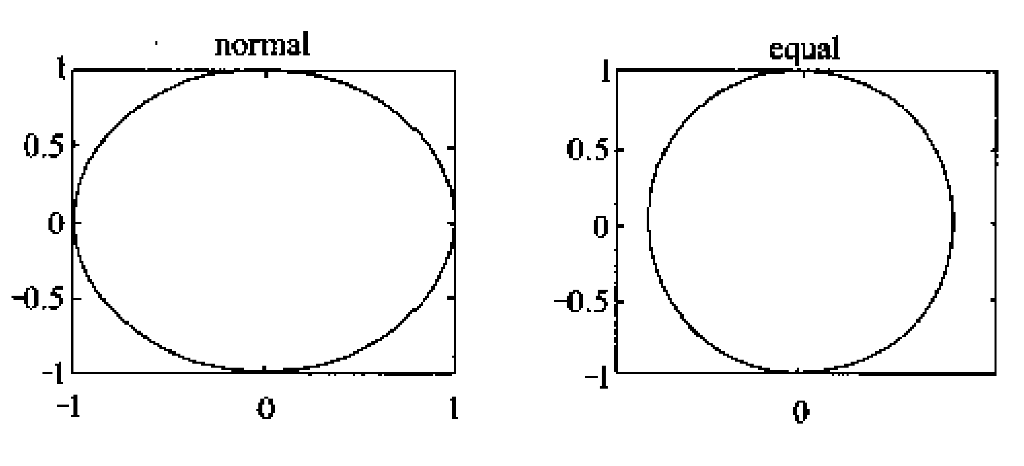


图 2.9　axis('equal')命令的作用

2.5.5　三维曲线和曲面

1. 空间曲线绘制——plot3

其用法大体与 plot 相仿。格式为：

```
plot3(x,y,z,'s').
```

其中 s 为线型颜色符。例如键入

```
z=0:0.1:4*pi;x=cos(z);y=sin(z); plot3(x, y, z)
```

得出的将是一条空间螺旋线，如图 2.10 所示。

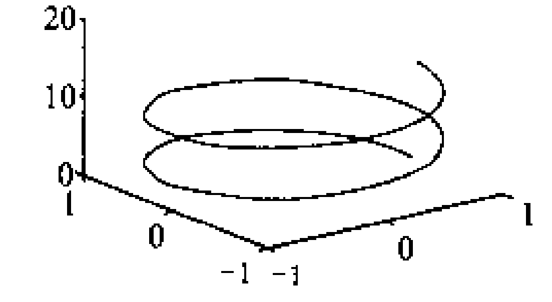


图 2.10　空间螺旋曲线

2. 空间曲面的绘制

函数 mesh 和 surf 用来绘制三维曲面。三维曲面方程应有 x，y 两个自变量，因此，先在 X-Y 平面上建立网格坐标，每一个网格点上的数据 Z 坐标就定义了曲面上的点。通过直线（mesh）或小平面（surf）连接相邻的点就构成了三维曲面。

mesh 函数可以把一个大矩阵形象化地表示出来。例如，函数 sinc（r）=sin（r）/r（其中 r 是 X-Y 平面上的向径）的立体图形是很生动的。可用下面的方法来绘制，程序如下：

```
x=-8:0.5:8;y=x';                        % 生成一维的自变量数组
X=ones(size(y))*x;Y=y*ones(size(x));    % 生成二维的自变量平面
R=sqrt(X.*X+Y.*Y);  z=sin(R)./R;        % 生成因变量
mesh(z), pause                          % 画三维曲面
```

第 1 行命令定义了函数计算的 x，y 取值范围，每一个方向有 33 个样本点；第 2 行命令建立了共有 33×33=1089 个网格点的坐标矩阵 X 和 Y，形成了 33×33 网格的矩阵；第 3 行程序表示数据点到原点的距离。并求得 sinc 函数值，最后用 mesh 函数绘出图形。

图 2.11 画出各点的 X 坐标，即竖线；画出各点的 Y 坐标，即横线。在所关心的 z 的定义域内列出 X，Y 后，即可进行函数的计算和绘图。MATLAB 也提供了生成网格点坐标的专用函数 meshgrid。用这个函数时，上述的两行程序可简化为一条语句：

```
[X,Y] = meshgrid(-8:0.5:8,-8:0.5:8);
```

执行了第 3 行程序后将得出以下的警告：

```
Warning: Divide by zero
```

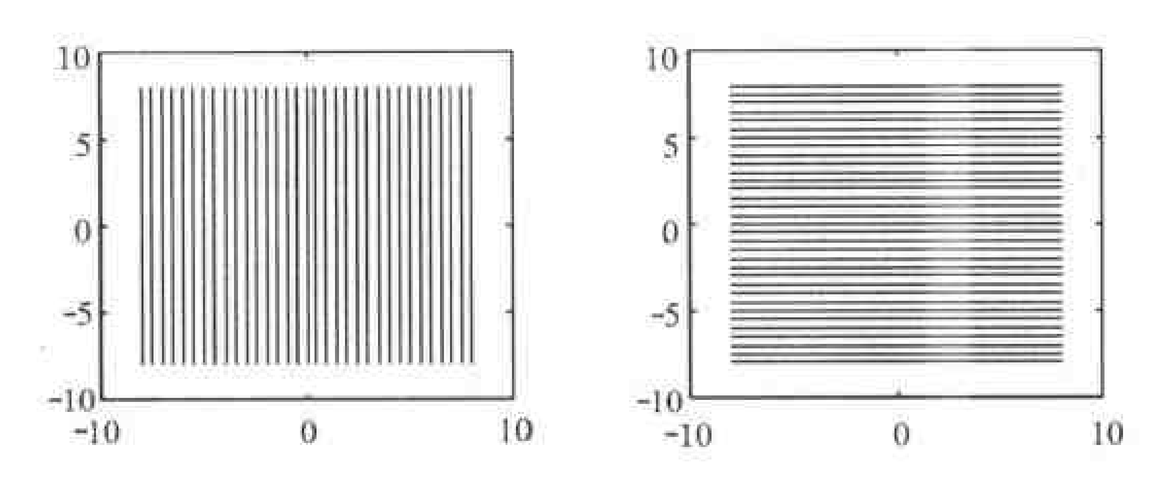


图 2.11　由 meshgrid 函数求出的 *X*，*Y* 矩阵对应的 *X*，*Y* 坐标

即出现了被零除的运算。实际上，这发生在 *R*=0（即原点）处，该处 sin（R）也为零，所以得到 NaN。产生的三维曲线见图 2.12，在 *R*=0 处缺掉一个点，因为 NaN 是无法画出的。从这里可以看出 IEEE 算法的优越性。如果照过去的方法，出现零做除数这种非法运算，就要退出系统，取消全部结果，那就得不到这个漂亮的图形，其实非法运算只是 1089 个点中的一个点。不能"攻其一点，不及其余"。解决这问题的方法也很简单，只要把样本点移离原点即可。为此把程序改为：

```
R=sqrt(X.*X+Y.*Y)+eps;  z=sin(R)./R;  figure(1), mesh(z)
```

即把原来的 *R* 值移动一个极小的数值 eps，运行就没有问题，而图上不再有缺掉的点了。把上式中的 *R* 改成 abs(x)+abs(y)（称为一范数，sqrt(x*x+y*y)称为二范数）。

键入

```
R=abs(X)+abs(Y)+eps;  z1=sin(R)./R;
figure(2),   surf(z1)
```

得出的三维曲面见图 2.13。

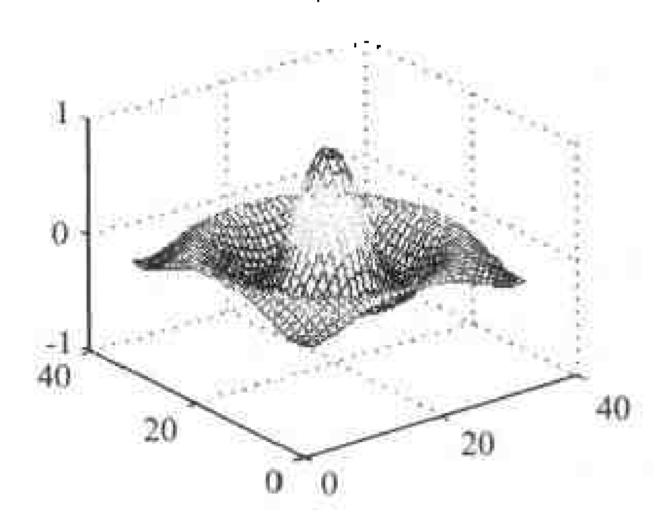


图 2.12　z=sin(R)./R 的三维曲面图

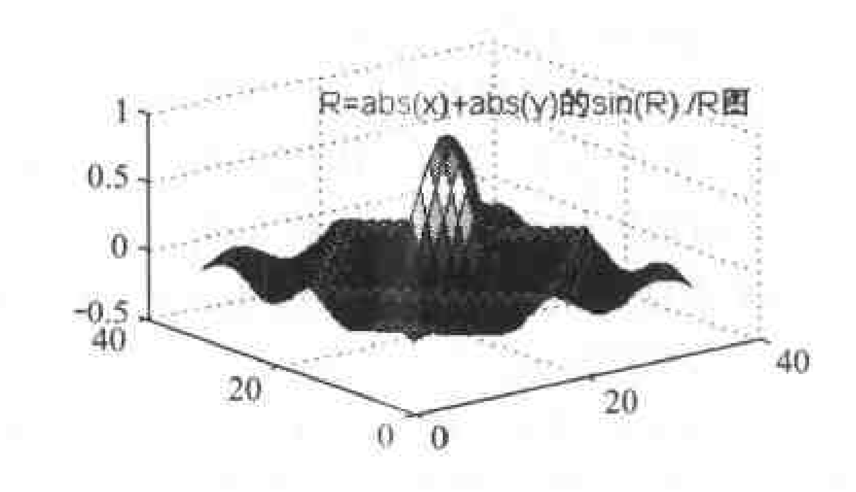


图 2.13　当 R 取一范数时 z=sin(R)./R 的三维曲面图

3. 其他三维图形绘制命令

三维图形绘制命令中有一组命令属于外观变换。

• view（方位角，俯仰角）可以变换立体图的视角，例如:

键入　`view(20, 0)`

就得到另一种三维图形，见图 2.14 中的第 4 子图（方位角，俯仰角的默认值为[37°，30°]）。

• shading flat 或 shading interp 可把曲面上的小格平滑掉，使曲面成为光滑表面。Shading faceted 是默认状态，它使曲面上有小格。MATLAB 5.x 新增了 rotate3d 命令，执行此命令后，用户可以用鼠标拖动立体图形作空间连续转动。

另有一组等高线绘制命令——contour 命令，把曲面的等高线投影在 X-Y 平面上，也就是普通地图中的画法。contour3 在三维立体图中画出等高线，这些等高线就像云那样浮在相应的高度上。下列程序使用了三维绘图库中的一些命令，其结果可从图 2.14 中看出。有关彩色的命令在后面讨论。

```
subplot(2, 2, 1), R=sqrt(X.^2+Y.*Y); z=sin(R)./R; meshc(z), pause
title('meshc(z), shading flat'), shading flat
subplot(2, 2, 2), R=sqrt(X.^2+Y.*Y)+eps; z=sin(R)./R;meshz(z), pause
title('meshz(z), shading interp'), shading interp
subplot(2, 2, 3), R=abs(X)+abs(Y)+eps; z1=sin(R)./R; surfc(z1), pause
title('surfc(z1), shading flat'), shading flat, %colormap(gray)
subplot(2, 2, 4); surfl(z1), view(20, 0)
title('surfl(z1), view(20, 0)')
```

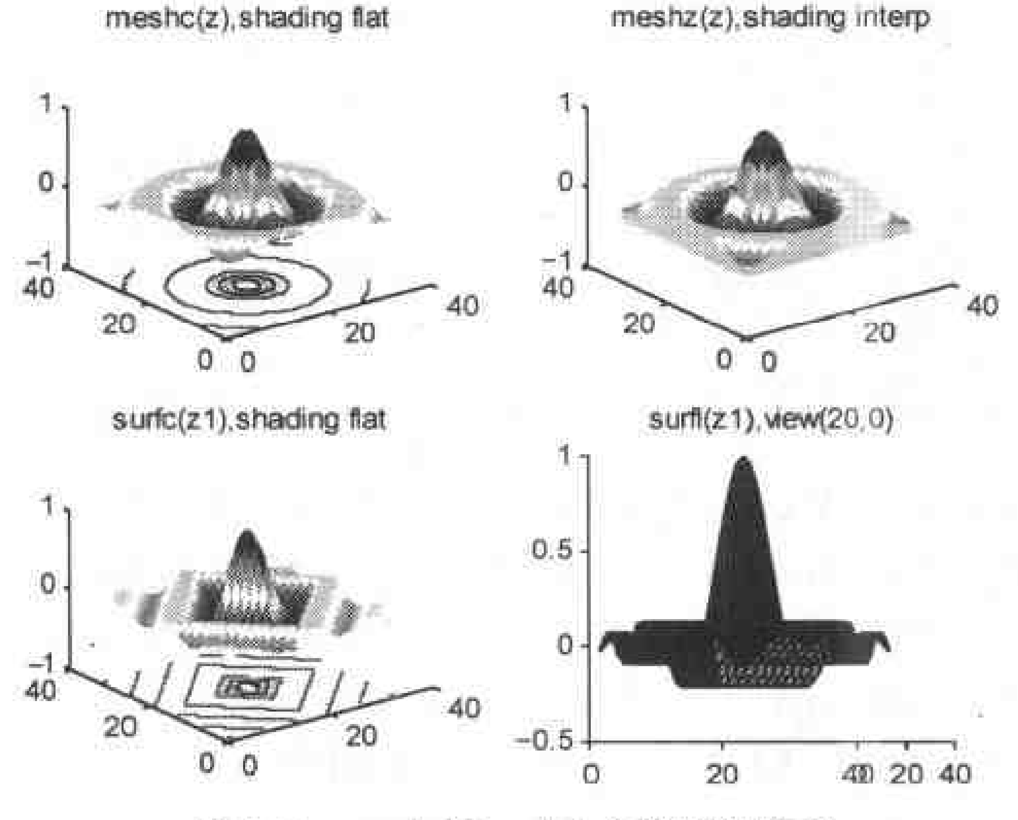


图 2.14　mesh 和 surf 命令的其他形式

2.5.6　特殊图形和动画

表 2.15 列出了 MATLAB 中的一些特殊图形函数，其中一部分是各种不同学科和领域中用到的特殊二维和三维图形。例如，前面提到过的 stem、stairs 等，pie 和 bar 是在管理科学中常用的饼图和条形图，compass 是电路中常用的相量图，在应用篇中还会介绍，读者可以自行试用。另一部分是前面已介绍过的等高线图形，此处不多占篇幅。

表 2.15　特殊图形库(specgraph) (u)

特殊的二维图形	area	填满绘图区域	feather	羽状图
	bar	条形图	fill	填满两维多边形
	barh	水平条形图	pareto	Pareto 图
	bar3	三维条形图	pie	饼图
	bar3h	三维水平条形图	plotmatrix	矩阵散布图
	compass	极坐标向量图	ribbon	画成三维中的色带
	comet	彗星轨迹图	stem	离散序列绘图
	errorbar	误差条图	stairs	阶梯图

续表

等高线图形	contour	等高线图	pcolor	伪彩色图.
	contourf	填充的等高线图	quiver	箭头图
	contour3	三维等高线图	voronoi	Voronoi 图
	clabel	等高线图标出字符		
特殊三维图形	comet3	三维彗星轨迹图	slice	实体切片图
	meshc	三维曲面与等高线组合图	surfc	三维曲面与等高线组合图
	meshz	带帘的三维曲面	trisurf	三角表面图
	pie3	三维饼图	trimesh	三角网状表面图
	stem3	三维 stem 图	waterfall	瀑布图
	quiver3	三维 quiver 图		
图像显示	image	显示图像	imread	从图形文件读出图像
	imagesc	缩放数据并做为图像显示	imwrite	把图像写入图形文件
	colormap	颜色查找表	imfinfo	关于图形文件的信息
电影和动画	capture	从屏幕抓取图形文件	rotate	绕给定方向旋转对象
	moviein	初始化电影帧存储器	frame2im	把电影帧转换为索引图像
	getframe	获取电影帧电影帧	im2frame	把索引图像转换为电影帧
	movie	重放录下的电影帧		
实体	cylinder	生成圆柱体	sphere	生成球体

在这里需要着重介绍的是 MATLAB 的动画命令。它总共只有三条命令：moviein，getframe 和 movie。用 getframe 把 MATLAB 产生的图形存储下来，每个图形成一个很大的列向量，再用 N 行这样的列保存 N 幅图，成为一个大矩阵。最后用 movie 命令把它们连起来重放，就可以产生动画效果。Moviein 用来预留存储空间以加快运行的速度（不用也可以）。下面是 MATLAB 手册上提供的一个漂亮的动画程序。

```
axis equal,    % 因为产生的图形是圆形，故把坐标设成相等比例
M = moviein(16);    % 为变量 M 预留 16 幅图的存储空间
    for j=1:16          % 作 16 次循环
        plot(fft(eye(j+16)));    % 画图
        M(:, j)=getframe;        % 依次存入 M 中
    end
```

运行完这几句程序后，16 幅画面（每幅用 16858*8=134864 个字节）就放在矩阵 M 中；再键入

```
movie(M, 30)
```

MATLAB 就把 M 中图形播放 30 遍。读者可自行在计算机上实践。

2.5.7　彩色、光照和图像

为了更好地显示图形，特别是空间图形，MATLAB 使用了彩色和光照。颜色提供了三维图形中的第四维坐标，扩展了图形表达的能力，光照又进一步改善了视觉效果，这也是 MATLAB 的一个重要特色。但因为彩色印刷的成本太高，在本书中不便展开，建议读者自己在计算机上实践。

在三维曲面绘图命令中，加入第四维变元，例如，mesh(x，y，z，w)，则 w 的大小就用颜色表示，即在坐标值为(x，y，z)的点上，赋予对应于 w 值的颜色。颜色与 w 值的一一对应关系，用彩色条（colorbar）来表示，用命令 colorbar 可得到这个关系，它将在已有的图形右侧，加上一条垂直的彩色条，并标以对应

的 w 值。

如果在三维曲面绘图命令中，没有第四维变元，则颜色轴将与 Z 轴一致，有一一对应的关系。例如：键入

```
[x, y]=meshgrid([-2:.2:2]);
z = x.*exp(-x.^2-y.^2);
surf(x, y, z), colorbar
```

所得图形及彩色条如图 2.15（a），可见 z 最大处有深红色，z 最小处为深蓝色。

如果把末行改成

```
surf(x, y, z, gradient(z)), colorbar
```

则彩色轴将表示曲面的梯度，也就是产生第四维的坐标，见图 2.15（b）。可以看出，在峰点谷点之间的斜率最大处有最深的颜色，负斜率处为深蓝色。

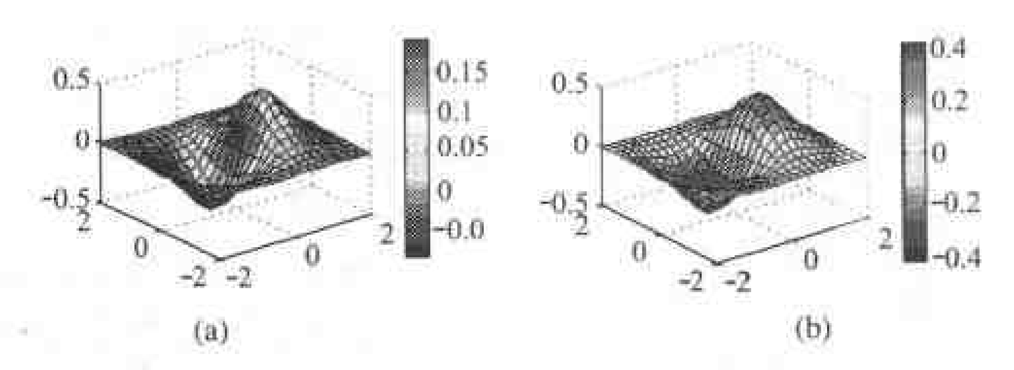


图 2.15 颜色坐标分别表示第三维(z)和第四维（梯度）

彩色条分为 64 个等级，其颜色排序的默认值为 hsv，但可由用户自行修改设定为 hot，cool，gray，copper，pink，jet，prism 等，其语句为 colormap(cool)。大家知道，任何颜色均可用红绿蓝三色的适当组合来产生。因此，有左端变量的语句 m=colormap 将返回一个 64×3 阶的矩阵。其各列依次表示红绿蓝三种基色的分量。rgbplot(m)则用红绿蓝三种曲线表示三基色分量的大小。表示颜色的另一种方法是色调-饱和度-亮值(hsv)方式，彩色电视机的调色就用这种方式。MATLAB 也提供了两者之间的转换函数。例如，对默认的彩色图 m 求 h=rgb2hsv(m)，则 h 也是一个 64×3 阶的矩阵，只不过它用的是 hsv 颜色表示方式，它的第二列、第三列全为 1，说明其饱和度及亮值均为 1，仅仅色调从 0 到 1 单调地变化，所以它才得到了 HSV 彩色图的名称。这个名称不太好，很容易与 hsv 的颜色表示方式相混淆。

把彩色条离散化形成伪彩色板，它默认地分为 64 份，键入 pcolor（[1:65;1:65]'）（65 个分割点产生 64 根彩条），可得出伪彩色板图，如图 2.16 所示。键入 hot(8)把伪彩色板设置为 hot 色并将其改为 8 份，显示这个伪彩色根应当用命令 pcolor([1:9;1:9]')。在自动坐标设定的情况下，彩色条应恰好覆盖图中的最小到最大的坐标值（例如，在代表 z 值时为[z_{min} z_{max}]）并略有余量。人工设定的命令为 caxis([cmin cmax])， 其作用是把彩色条中的最小值、最大值分别与 cmin、cmax 相对应，以达到用户特定的显示需要。

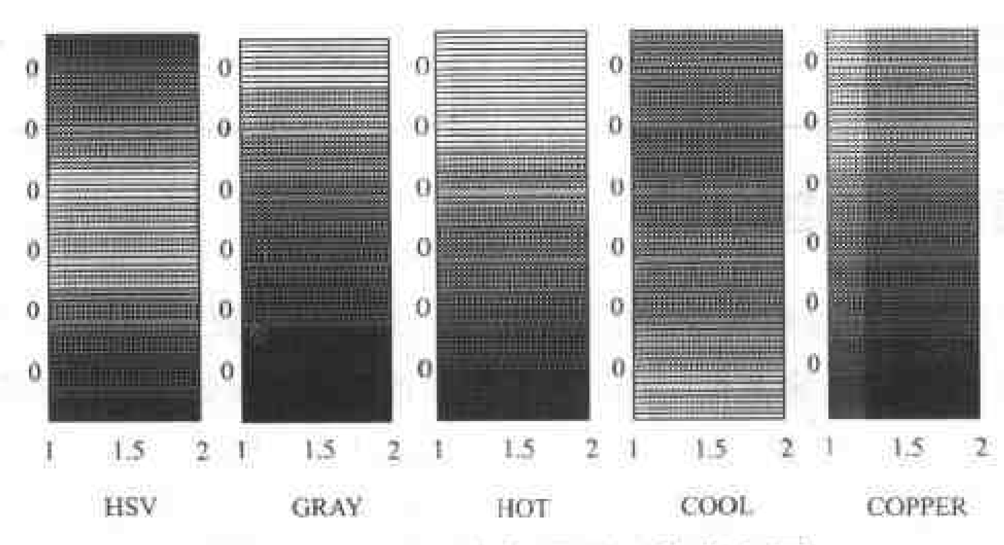


图 2.16 不同彩色设置下的彩色图

在 MATLAB 4.x 中专门设有颜色和照明函数库，在 MATLAB 5.x 中，它被并入三维图形库中，见表 2.16。其中光照模型诸函数的输入变元中都要规定光源的方向或坐标，例如 surfl(peaks，[30，0])可得出光源在方位角 30°、俯仰角 0° 方向照射时的 peak 曲面。读者可根据应用的需要自行试验其用法，从 help 文本中可以获得相应的应用信息。

图像和图形的不同之处在于：图形只有黑白两色，而图像则有深浅之别，称为灰度，彩色图像还有色度。通常黑白图像有 256 个灰度等级，因此，图上的每个点要用 8 位二进制表示。如果 ***A*** 是一个 *M*×*N* 阶的矩阵，其每个元素都是 8 位二进制的整数（0~255），则 image(A)就创建了一幅 *M*×*N* 元素组成的图像，其每个元素按其值在彩色图（colormap）中取色，键入 image(A)，colormap(gray)将得出黑白图像。

表 2.16　三维绘图和光照函数库(graph3d) (q)

绘制三维曲线命令	plot3	在三维空间中画线和点	mesh	三维网格图
	fill3	在三维空间中绘制填充多边形	surf	三维曲面图
颜色控制	colormap	彩色查询表	caxis	伪彩色坐标轴定标
	shading	彩色阴影方式	hidden	消隐或显示被遮挡的线条
	brighten	改变彩色图的亮度		
彩色图	hsv	色调-饱和度-亮值彩色图	gray	线性灰度彩色图
	hot	黑-红-黄-白彩色图	cool	蓝绿和洋红阴影彩色图
	bone	蓝色色调的灰度彩色图	copper	铜色调的线性彩色图
	pink	线性粉红色阴影彩色图	prism	光谱彩色图
	jet	HSV 彩色图的变型	flag	红、白、蓝、黑交互的彩色图
	spring	品红和黄阴影彩色图	summer	绿和黄阴影彩色图
	autumn	红和黄阴影彩色图	winter	蓝和绿阴影彩色图
	white	全白彩色图	lines	带颜色线的彩色图
	colorcube*	增强的彩色立方体彩色图	colstyle	从字符串分解出颜色和字体
彩色图有关的函数	colorbar	显示彩色条	hsv2rgb	由 hsv 向红绿蓝(rgb)转换
	rgb2hsv	红绿蓝向 hsv 转换	contrast	变灰度图为对比增强方式
	rgbplot	用 rgb 绘彩色图	spinmap	旋转彩色图
视点控制	view	规定三维图的视点	viewmtx	视点变换矩阵
	rotate3d *	用鼠标拖动图形作三维旋转		
照明模型	surfl	带照明的三维曲面图	specular	镜面反射
	lighting	光照模式	material *	材料反射模式
	diffuse	漫反射	surfnorm	曲面法线
轴系控制	见二维图形函数库(表 2.14)，增加了 zlabel 等			
图形标注	见二维图形函数库			
打印输出	见二维图形函数库			

2.5.8　低层图形屏幕控制功能

直到现在，所学的都是高层绘图命令。绘图本来是一件很复杂繁琐的工作，MATLAB 怎么把它简化的呢，主要是它代用户做了大量的事务性工作。有些是有根据的，例如，坐标范围根据输入数据的最大最小值来选择；有许多则没有一定的道理，只是开发人员任意给出的默认值。例如，图形的背景色、绘图线条的线宽、线型、颜色、坐标网格的密度及其标注尺寸等等。在入门的时候，这些内容无需关心，最好由

软件开发者帮助设好，免得分散注意力。但一旦遇到默认值不满足实际应用中的特殊需要时，想修改其中某些参数，就会感到高层命令太自动化智能化了，不便于人的干预，低层图形屏幕控制功能就是为补充这个不足的。

MATLAB 中的图形对象共有 10 类，用图 2.17 表示。

root（根对象）在最上层，其子类只有一种 figure（图形对象），但可有多个，下面有 3 个子类，即 uicontrols，uimenus，axes；其中前两类是生成专用人机界面的，这里不涉及，只考虑 axes（坐标系对象），一个图也可有多个坐标系（如 subplot），在它之下，有 6 个子类，即 text（图中文字），line（线），patches（块），surface（曲面）和 image（图像）。任何图形都是由这 9 类图形对象组成的。figure 是 axes 的父类，而 axes 又是其他 6 种对象的父类。执行 figure，plot，mesh，surf，fill，text 等高层命令时，MATLAB 将生成某些图形对象，同时，也生成了一个称为“图形句柄”的数组，其中包含了产生各个对象所需的各种默认参数。修改其中的参数就可改变当前对象中的图形特性。

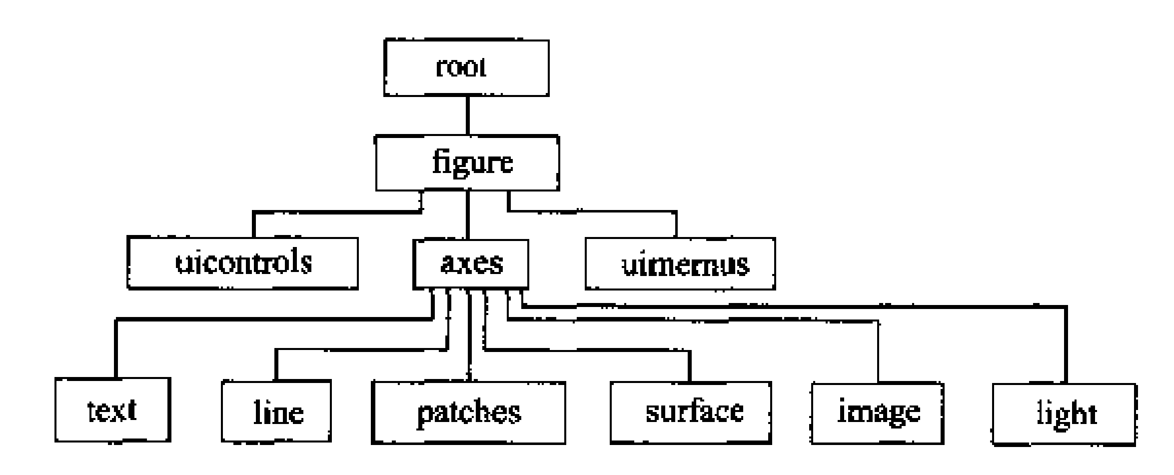


图 2.17 MATLAB 中图形对象的分类和层次关系

（1）取得图形句柄的方法

高层图形命令 figure、plot、mesh、surf 等都是无左端变量的。如果在其左端放一个变量，则执行该绘图语句后，该变量就成为此图形的“句柄”名。用 get 命令可以得到它的内容。

例如键入 `surh=surfc(peaks);`

其结果为
```
surh = 88.0001
       93.0001
       ...
       100.0001
```

这表明 surfc 命令产生了 9 个对象。因为 surfc 除产生三维曲面外还产生了 8 根等高线。其编号分别为 88.0001，93.0001，94.0001，95.0001，96.0001，97.0001，98.0001，99.0001，100.0001，这些编号只有内部的意义，对用户来说，主要记住句柄名，把它们命名为 surh(1)，…，surh(9)即可。每一个对象都有自己的句柄。例如：

键入
```
get(surh(1))
```
得
```
CData = [ (49 by 49) ]       注：颜色数据
EdgeColor = flat                 边缘颜色
EraseMode = normal               可擦模式
FaceColor = [0 0 0]              前景颜色
LineStyle = -                    线型
LineWidth = [0.5]                线宽
```

```
    MarkerSize = [6]                    标志点尺寸
    MeshStyle = both                    网格型式
    XData = [ (1 by 49) ]               X 数据
    YData = [ (49 by 1) ]               Y 数据
    ZData = [ (49 by 49) ]              Z 数据
……
    Children = [ ]                      子类对象的句柄
    Parent = [56.0001]                  父类句柄编号
    Type = surface                      对象类型
```

这里删略了某些无关的句柄内容，在入门课程中，不可能涉及太深。用 gca 命令可得到当前 axes（即上述对象的父类）句柄，用 gcf 命令可得到当前的 figure（即上述对象的祖类）句柄，它们的内容更多，此处从略。

（2）修改句柄内容的方法(set 命令)

```
set(surh(1), 'linewidth', 2, 'markersize', 3)
```

将把三维曲面中的线宽加粗到 2 毫米，标志点直径减小为 3 毫米。但图中的等高线则不受影响。

（3）修改厂设默认参数的方法

前面介绍的方法，只能一张图一张图，一根曲线一根曲线地去修改，如果希望把每张图上的曲线都改粗一些，最好一进 MATLAB 环境就把公司出厂设定的默认值改掉。用下述命令可以做到。

```
set(0, 'DefaultLineLineWidth', 1.5)
```

0 代表根对象，把它改了，则它所有的子类和孙类的曲线线宽也都改了。如果只需把当前图中的全部线型改变为点划线，可用

```
set(gcf, 'DefaultLineLineStyle', '-.' )
```

可更改的厂设参数很多，如 Defaultfigurecolor，Defaultsurfaceedgecolor，DefaultAxesColor，DefaultAxesColorOrder，DefaultAxesLineStyleOrder，DefaultTextRotation 等等，此处无法穷尽，读者可参阅 MATLAB 有关手册。

2.6 M 文件及程序调试

在入门阶段，通常在行命令模式下工作。键入一行命令后，让系统立即执行该命令。用这种方法时，程序可读性很差且难以存储。对于复杂的问题就应编成可存储的程序文本，再让 MATLAB 执行该程序文件，这种工作模式称为程序文件模式。

由 MATLAB 语句构成的程序文件称为M文件，它将 m 作为文件的扩展名。例如：文件 expml.m 用来计算矩阵指数函数的值。因为它是 ASCII 码文本文件，所以可以直接阅读并用任何编辑器来建立。

M文件可分为两种：一种是主程序，也称为主程序文件（script file），是由用户为解决特定的问题而编制的；另一种是子程序，也称为函数文件（function file），它必须由其他 M 文件来调用。函数文件往往具有一定的通用性，并且可以进行递归调用（即自己可以调用自己）。MATLAB 的基础部分中已有了 700 多个函数文件，它的工具箱中还有千余个函数文件，并在不断扩充积累。MATLAB 软件的大部分功能都是来自其建立的函数集，利用这些函数可以使用户方便地解决他们的特定问题。

2.6.1 主程序文件

主程序文件的格式特征如下：

（1）用 clear、close all 等语句开始，清除掉工作空间中原有的变量和图形，以避免其他已执行的程序残留数据对本程序的影响。

前几行通常是对此程序用途的说明，特别是在运行时对用户输入数据的要求，更要叙述清楚。不然别人就看不懂也用不成，连自己日后也会遗忘。这些注释行必须以%号开始，以便计算机执行时不予理会。MATLAB 规定，在键入“help 文件名”时，屏幕上会将该文件中以%起头的最前面几行的内容显示出来，使用户知道如何使用。这些注释是可以用汉字的。%号也可以放在程序行的后面做注释，MATLAB 将不执行该字符后的任何内容。

（2）程序的主体。如果文件中有全局变量，即在子程序中与主程序共用的变量，应在程序的起始部分注明。其语句是

➢ global 变量名1 变量名2 ……

为了改善可读性，要注意流程控制语句的缩进及与 end 的对应关系。另外，程序中必须都用半角英文字母和符号（只有引号括住的和%号后的内容可用汉字）。特别要注意英文和汉字的有些标点符号（如句号、冒号、逗号、分号、引号乃至%、=、() 等），看起来很相似，其实代码不同。用错了，不但程序执行不通，而且几乎必定死机。因此键入程序时，最好从头到尾用英文，不要插入汉字。汉字可在程序调试完毕后加入。用 MATLAB 5.x 的编辑器比较好，因为它对出现的非法字符会显示出鲜明的红色，引起用户的注意。且在它的菜单 View 中，选 Auto Indent Selection 项可以自动对程序进行缩进排版。

（3）整个程序应按 MATLAB 标识符的要求起文件名，并加上后缀 m。用 MATLAB 4.x 时，文件名长度不要超过8个字符。文件名中不允许用汉字，因为这个文件名，也就是MATLAB 的调用命令，它是不认汉字的。将文件存入自己确定的子目录中。该子目录应置于 MATLAB 的搜索路径下。所以还应避免出现汉字路径名。

这样就完成了程序的编制。在 MATLAB 的命令窗中键入此程序的文件名后，系统就开始执行文件中的程序，主程序文件中的语句将对工作空间中的所有数据进行运算操作。

【例 2.6】 要求列写一个求 Fibonnaci 数的程序。它是一个数列，从[1，1]开始，由数列的最后两个元素之和生成新的元素，依次递推。其程序如下：

```
% 计算 Fibonnaci 数的 M 文件
clear, close all
N = input('输入最大数值范围 N=')
f = [1, 1]; i = 1;          % 变量的初始化
while f(i)+f(i+1)<N         % 循环条件检验
f(i+2) = f(i+1)+f(i); i=i+1; % 求 Fibonnaci 数的算式
end,
f, plot(f)          % 显示和绘图
```

将此程序以文件名 fibon.m 存入某一 MATLAB 搜索目录下。在 MATLAB 命令窗中键入 fibon，系统就开始执行这个程序。它首先会要求用户输入 N，然后计算数值小于 N 的 Fibonnaci 数，并绘出图线。设输入 N＝100，得出（图线略）：

```
f = 1    1    2    3    5    8    13    21    34    55    89
```

该文件执行结束，变量 f 和 i 仍保存在工作空间中。

【例 2.7】 列出求素数的程序。所谓素数就是只能被它自身和 1 除净的数。程序如下：

```
% 求素数(prime number)的程序
clear, close all
N =input('N= '),   x=2:N;                    % 列出从 2 到 N 的全部自然数
for u=2:sqrt(N)                              % 依次列出除数（最大到 N 的平方根）
    n=find(rem(x, u)==0 & x~=u);             % 找到能被 u 除净而 u 不等于 x 的数序号
x(n)=[];                                     % 剔除该数
end, x                                       % 循环结束，显示结果
```

以 prime.m 为名存入系统，就可以执行了。给出 N=40，结果为

```
x =   2    3    5    7   11   13   17   19   23   29   31   37
```

2.6.2　人机交互命令

在执行主程序文件中，往往还希望在适当的地方对程序的运行进行观察或干预，这时就需要人机交互的命令，调试程序时，人机交互命令更不可少。这些命令可以 MATLAB 语言结构库（见表 2.17）中调用。下面介绍几条：

· echo on(off)　一般情况下，M 文件中的命令不会显示在屏幕上。而在命令 echo on 之后，会在执行每行程序前先显示其内容。

· pause(n)　程序执行到此处，暂停 n 秒，再继续执行。如果没有括号参数，则等待用户键入任意键后才继续执行。

· keyboard　程序执行到此处暂停，在屏幕上显示字符 K，并把程序的输入和执行权交给用户（键盘）。用户可以像在普通 MATLAB 命令窗中那样进行任何操作（例如，检查中间结果等）。如果需要系统恢复运行原来的程序，只需键入字符串 return。在 M 文件中设置该命令，有利于进行程序调试以及临时修改变量内容。

表 2.17　语言结构库(lang) (k)

	名　称	功　能	名　称	功　能
估值并执行	eval	执行 MATLAB 语句字符串	feval	执行由字符串命名的函数
	evalin *	估值工作空间中的表示式	builtin	从超载方法执行内置函数
	assignin *	分配工作空间中的变量	run	运行程序文件
流程控制语句	if	条件执行命令	else	与 if 联用
	elseif	与 if 联用	end	For，while，if 语句的终点
	for	确定次数的重复语句	while	非确定次数的重复语句
	break	终止执行循环	return	返回到调用函数
	switch *	在表示式的几种情况中选择	otherwise *	switch 语句中的默认值
	case *	switch 语句中的情况		
程序、函数和变量	script	MATLAB 程序文件——M 文件	function	加入新函数
	global	定义全局变量	mfilename *	当前执行的文件名
	list	以逗号分割的清单	isglobal	是全局变量是为真
	exit	检查变量或函数是否存在		

续表

	名　　称	功　　能	名　　称	功　　能
变元管理	nargchk	检验输入变元的数目	nargin	输入变元的数目
	nargout	输出变元的数目	varargin *	长度可变的输入变元清单
	varargout *	长度可变的输出变元清单	inputname *	输入变元的名称
信息显示	error	跳出函数并显示信息	lasterr	最近的出错信息
	warning *	显示警告信息	errortrap *	在测试中跳过错误
	disp	显示数组	fprintf	显示格式化信息
	sprintf	把格式化数据写成字符串	echo	显示执行的 MATLAB 语句
人机交互命令	input	提示用户输入	keyboard	调用等待键盘输入
	menu	生成用户输入的选择菜单	pause	暂停，等待用户响应

· input（'提示符'）　程序执行到此处暂停，在屏幕上显示引号中的字符串。要求用户输入数据。如程序为 X=input('X=')，则屏幕上显示 X=，输入的数据将赋值给 X。数据输入后，程序继续运行。input 命令也可以接受字符，其格式为 Y=input（'提示符'，'s'），此时 Y 将等于输入的字符串。

· ^C　强行停止程序运行的命令。^C 念作（Control-C），即先按下 Ctrl 键，不抬起再按 C 键。在发现程序运行有错，运行时间太长时，可用此方法中途终止它。

· menu　是用来产生人机交互备选择菜单的命令，参阅 help 文件。

2.6.3 函数文件

函数文件是用来定义子程序的。它与程序文件的主要区别有 3 点：

· 由 function 起头，后跟的函数名必须与文件名同；

· 有输入输出变元（变量），可进行变量传递；

· 除非用 global 声明，程序中的变量均为局部变量，不保存在工作空间中。

先看一个简单的函数文件，其文件名 mean.m。

键入

```
type mean
```

得到

```
function  y = mean(x)
% MEAN 求平均值。对于向量，mean(x)返回该向量 x 中各元素的平均值
% 对于矩阵，mean(x)是一个包含各列元素平均值的行向量
[m, n] = size(x);
     if m==1 M=n;    end      % 处理单行向量
y=sum(x)/m
```

文件的第一条语句定义了函数名、输入变元以及输出变元。没有这条语句，该文件就成为程序文件，而不再是函数文件。输入变元和输出变元都可以有若干个，但必须在第一条语句中明确地列出。

程序中的前几条带%的字符行为文件提供注解，键入 help mean 命令后，系统将显示这几条文字，作为对文件 mean.m 的说明，这和主程序文件相同。

变量 m，n 和 y 都是函数 mean 的局部变量，当 mean.m 文件执行完毕，这些变量值会自动消失，不保存在工作空间中。如果在该文件执行前，工作空间中已经有同名的变量，系统会把两者看做各自无关的变量，不会混淆。这样，调用子程序时就不必考虑其中的变量与程序变量冲突的问题了。如果希望把两者看成同一变量，则必须在主程序和子程序中都加入 global 语句，对此共同变量进行声明。

给输入变元 x 赋值时，应把 x 代换成主程序中的已知变量，假如它是一个已知向量或矩阵 **Z**，可写成 mean(Z)，该变量 Z 通过变元替换传递给 mean 函数后，在子程序内，它就变成了局部变量 x。

下面的例子是多输入变量函数 logspace，用于生成等比分割的数组。

```
function y = logspace(d1, d2, n)
%    LOGSPACE 对数均分数组
%    LOGSPACE(d1, d2)在 10^d1 与 10^d2 之间生成长度为 50 的对数均分数组
%    如果 d2 为 pi，  则这些点在 10^d1 和 pi 之间
%    LOGSPACE(d1, d2, n)生成的数组长度为 n, n 的缺省值为 50
if nargin == 2    n = 50;  end       % 输入变元分析及 n 的缺省值设置
if d2 == pi   d2 = log10(pi);end        % d2 为 pi 时的设置
y = (10).^ [d1+(0:n-2)*(d2-d1)/(n-1), d2]; % 将结果返回到输出变元
```

在本例中使用了特定变量 nargin 表示输入变元的数目。当只有两个输入变元时，它默认地设定 n=50，nargout 表示输出变元数目的变量。MATLAB 常常根据 nargin 和 nargout 的数目不同而调用不同的程序段，从而体现它的智能作用。

再来看 MATLAB 如何定义一个任意非线性函数的。在对微分方程进行数值积分或解任意非线性方程求解时，都需要先列出一个这样的函数文件。一个典型程序（文件名为 humps.m）如下：

```
function y = humps(x)
%HUMPS    由 QUADDEMO,   ZERODEMO 和 FPLOTDEMO 等程序调用的一个函数
%    HUMPS(X) 是一个在 x = 0.3 and x =0.9 附近有尖锐极大值的函数
%    参看 QUADDEMO,   ZERODEMO 和 FPLOTDEMO
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

程序中的运算都采用元素群算法，以保证此函数可按元素群调用。MATLAB 中几乎所有的函数都能用元素群运算，所以，本书中自编的子程序，也尽量满足元素群算法的要求。

2.6.4 文件编辑器及程序调试

用 MATLAB 4.x 时，建议用 Windows 中的 notepad 作为其程序编辑器。如果用 MATLAB 5.x，则用 MATLAB 提供的编辑器较为理想，它把编辑与调试结合在一起了。实际上，MATLAB 的主程序是比较好调试的，因为 MATLAB 的查错能力很强，配上工作空间中变量的保存和显示功能，不需要用专门的调试命令，调试也可以很方便地进行。

需要用调试命令的主要是函数程序。因为在函数程序出错而停机时，其变量不被保存。虽然它也会指出出错的语句，但因为子程序中的变量在程序执行完毕后会自动消失，其他现场数据都无记录。会给调试带来很大困难。解决这个问题可采用下列措施。

- 把某些分号改为逗号，使中间结果能显示在屏幕上，作为查错的依据。
- 在子程序中适当部位加 keyboard 命令。此处，系统会暂停而等待用户键入命令。这时子程序中的变量还存在于工作空间中，可以对它们进行检查。
- 将函数文件的第一行前加%号，使它成为程序文件，进行初步调试。第一行中的输入变元，可改用 input 或赋值语句来输入，调好后再改回函数文件。
- 使用 MATLAB 提供的调试命令。其命令共有 11 条，见表 3.1。根据经验，当程序不太长（例如 20 行以下）时，用调试命令反而麻烦。所以，在入门课程中不做介绍。

第3章　MATLAB 的开发环境和工具

作为一种优秀的计算软件，MATLAB 之所以能够成功，不仅因为其语法和编程的简洁高效，而且在于它有一个便于使用开发并与其他软件（甚至硬件）进行交互通信的环境。比如它可以在多种计算机机型和操作系统下运行，其使用的界面和程序又完全相同，使程序设计与平台无关；它能够和一些重要的文字编辑器及图形编辑器进行交互，把计算结果很方便地组织成为图文并茂的文章等等。从 MATLAB3.x 起，其语言已经相当成熟，版本的每一次升级，在语言方面的变化很少，变动主要集中在工具箱的扩充和开发环境的改善，这方面的内容涉及面很广，不可能在一本教科书中叙述清楚。本章将以 MATLAB 5.3 和 MATLAB 6.0 版本为主要对象，对此做一概要介绍。

3.1　MATLAB 与其他软件的接口关系

3.1.1　与磁盘操作系统的接口关系

1．变量的存储和下载

save 命令把工作空间中的全部变量值存入磁盘。其默认的文件名是 matlab.mat。第二次再用 save 命令时，如果仍用默认文件名，则原来文件中的数据就被冲销。如果只要把 a，b，c 三个变量保存在名为 aa.mat 的文件中，则可键入：

save aa a b c

mat 格式是人读不懂的，如果要存成 ASCII 码格式，则应再加上一个格式说明符：

save aa a b c –ascii

load 是 save 的逆过程，它把磁盘上存储的 mat 数据文件取回到 MATLAB 工作空间中。其默认的文件名也是 matlab.mat。在不用默认文件或默认格式时，其命令格式与 save 命令相仿，惟一的差别是它不能选择变量。例如 load aa，它把 aa.mat 文件中的全部数据连同其变量名都下载到工作空间中。

格式说明符还有多种，MATLAB 5.x 的默认格式与 MATLAB 4.x 又不同。因此，在 MATLAB 4.x 下存入的 mat 格式变量不能为 MATLAB 5.x 直接读出，必须在读命令的后面加上特殊的格式说明-v4，例如 load aa –v4，读者在遇到此问题时可从 help save 或 help load 中寻找详细说明。

表 3.1 列出了 MATLAB 的通用命令。

表 3.1　通用命令库(general) (I)

函数的管理命令	what	列出 M、MAT 和 MEX 文件	which	找函数和文件所在的子目录
	type	显示 M 文件的全部内容	pcode *	建立微码文件（P 文件）
	edit	编辑 M 文件	inmem	列出内存中的函数
	lookfor	在求助文字中搜索关键词	mex	编译 MEX 函数

续表

通用信息	help	在线帮助文件	whatsnew	未列入说明书的新功能信息
	helpwin	有独立视窗的在线帮助文件	readme	显示 readme 文件
	helpdesk	超文本帮助文件	ver	MATLAB 和各工具箱的版本
	demo	运行演示程序		
工作空间管理	who	列出工作空间变量	save	从工作空间存储变量到磁盘
	whos	列出工作空间变量详情	clear	从内存中清除变量和函数
	load	从磁盘取出变量到工作空间	pack	紧缩工作空间内存
管理搜索路径	path	查找和改变 MATLAB 搜索路径	rmpath *	在搜索路径上去除子目录
	addpath *	在搜索路径上增加子目录	editpath *	修改搜索路径
文件操作系统	cd	更改当前工作目录区	pwd	显示当前工作目录
	dir	列出子目录	web *	打开 Web 浏览器
	delete	删除文件	computer	当前的计算机型号
	getenv	获取环境参数	Ctrl C	中断 MATLAB 的运行
命令窗控制	profile	设置 M 文件执行时间	format	设置输出格式
	clc	清除命令窗	diary	保存 MATLAB 运行文字记录
	home	使光标复原到左上角	more	在命令窗中控制分页输出
启动退出	quit	退出 MATLAB	matlabrc	起动的主 M 文件
	startup	起动 MATLAB 时的 M 文件		
公共信息	info	关于 Mathworks 公司的信息	hostid	MATLAB 服务主顾的识别码
	subscribe	订购 MATLAB 须知		
调试命令	dbstop	设置断点	dbclear	清除断点
	dbcont	继续执行	dbdown	改变局部工作空间内容
	dbstack	列出谁调用谁的清单	dbstatus	列出所有断点的清单
	dbstep	执行一行或几行	dbtype	列出带行号的 M 文件
	dbup	改变局部工作空间内容	dbquit	退出调试模式
	dbmex *	调试 MEX 文件	dbstop	停止调试

2. 工作日志的记录

diary 命令可把 MATLAB 工作过程中的全部屏幕文字和数据以文本方式记录下来，成为一个工作记录，默认的文件名为 diary。因为它是文本文件，并可由任何文字处理器来修改编辑，所以有很大的使用价值，其用法如下。

当准备做记录时，在命令窗中键入 diary on 或 diary bbb，后者用 bbb.txt 为文件名。从此时开始，所有在 MATLAB 命令窗中出现的文字和数据都将记录在 diary.txt 或 bbb.txt 文件中。当需记录的过程结束，应键入 diary off。此后的屏幕内容即不做记录。如果再次使用 diary on 或 diary 文件名，则新记录的内容将接在原记录的后面，不会冲销原记录。diary 文件可以用 Notepad 或 WinWord 打开阅读。

为了避免在日志文件中记录不必要的调试过程和“垃圾内容”，应该在程序调试成功、运行无误后再打开日志文件，让程序正式运行一次。有时还需先键入 echo on，使得被执行的语句也在屏幕上显示并被记录到日志中去。记录中如发现有不必要的内容，可用文字处理器予以删改。diary 文件不能记录 MATLAB 运行中生成的图形。

3．日期和时间命令

MATLAB 中的某些命令是与操作系统有内在联系的。除了前面说过的它可直接应用的操作系统命令 dir、delete、cd 等之外，有关时间和日期方面的命令，都是从操作系统中提取数据的。这些命令在 MATLAB 4.2 中共有 6 个，即 clock、cputime、etime、date、tic、toc。表 3.2 中带*号的则是 MATLAB 5.x 中增补的。

表 3.2　时间和日期函数库(time fun) (w)

当前日期	now *	当前日期和时间的时间数	clock	当前日期的日期向量
	date	当前日期的字符串		
基本函数	datenum *	成序列的日期数	datevec *	日期向量
	datestr *	日期的字符串格式		
日期函数	calender	日历	eomday *	月末日的星期数
	weekday *	星期数	datetick *	日期的格式设定
定时函数	cputime	以秒计的 CPU 时间	etime	经历时间
	tic,toc	秒表定时器的启动和停止	pause	暂停等待时间

下面介绍如何确定作某种计算所需的时间。例如，想看看做 1 个 100×100 阶矩阵的求逆运算所需的时间，可以用下列 3 组语句之一。

- t0 =clock; y=inv(rand(100,100)); etime(clock, t0)
- t=cputime; y=inv(rand(100,100)); cputime-t
- tic; y=inv(rand(100,100)); toc

这 3 种方法的差别在于，第 1 种方法要先后两次提取年月日时分秒数据并将他们相减；第 2 种方法以开机时间为基准；第 3 种方法则用 tic 把秒表置零，求得的 toc 就是经历的时间。

4．不退出 MATLAB 环境运行其他软件

以"！"开始的命令表示这是一个 dos 操作系统的命令。可以用这个方法在不退出 MATLAB 环境的条件下，运行以 dos 操作系统为基础的其他软件。

3.1.2　与文字处理系统 WinWord 的关系

1．利用剪贴板进行交互

MATLAB 的程序要利用文字处理系统来编辑修改，它的运行结果（包括数据和图形）需要由图文处理系统来整理加工。因此它与 Word 图文处理系统有非常紧密的关系。它的命令窗中的所有文字数据及图形窗中的所有图形都可用 Windows 的剪贴板（Clipboard）送到 Word 中去，并可以用 Word 对它们进行编辑，形成图文并茂的书面报告。

在图形窗中截取图形时，应先用鼠标拖动边缘的方法将图形窗调到需要的大小，然后用鼠标单击菜单中的【Edit】项，在【Copy Options】子项中有【Metafile】(矢量模式）和【Bitmap】(点阵模式)。通常应选【Metafile】，因为这种模式便于在 Word 中作进一步的缩放修改。在设定完毕后，再选定【Copy】，图就放到剪贴板上去了。然后，可把这个图贴向 Word 的任何文本文件并在其中作进一步的编辑修改。在 MATLAB 中缩放可以保持图中标

注文字的可读性，而在 Word 中缩放图形往往会使文字排列不好。所以，建议在 MATLAB 中先把图形比例取得大体合适，避免到 Word 中作大幅度的缩放调整。

2. 文字编辑器的使用

在 MATLAB 5.x 中，已经把 Word 中的文字编辑功能集成为 MATLAB 的程序编辑和调试器。在图 1.2 显示的命令窗中，按下最左边的按钮，就会激活其程序编辑和调试器，生成图中的视窗。该视窗中的各个按钮的形式和功能与 Word 界面的几乎完全相同，所以不必细说。它的特殊之处在于：

（1）它会用不同颜色显示 MATLAB 规定的保留字（蓝）、非法字符（鲜红）、注释字符（绿）、引用字符（深红）等；

（2）存储文件名的后缀为.m——即生成的是 M 文件；

（3）当被编辑的文件以 function 开头，即被编辑的是一个函数文件时，MATLAB 5.3 会自动将存储文件名定为该程序中的函数名（见第 2.6 节中函数文件的命名规定）。

（4）能对程序自动缩进排版，便于阅读和调试。选定需要排版的程序段，单击菜单项【View】下的子项【Auto Indent Selection】，即可完成。

（5）它有程序调试器功能，反映在菜单项【Debug】的各子项中。

3. Notebook 软件工具

Notebook 是 Mathworks 公司开发的软件，它在 Word 和 MATLAB 两个软件系统之间搭起了一座双向接口的桥梁。当这个软件工作时，可在 Word 中输入含有部分 MATLAB 语句的文本文件。以后只要选中这些语句，再键入 Ctrl-Enter，该软件就会把这些语句送给 MATLAB 去执行，然后把运行的结果又送回 Word，并用不同的颜色显示输出和输入的不同。利用这个工具，教师可以边写教案，边检验教案中的程序语句。科技工作者也可一边写论文，一边让论文中的程序运行结果直接出现在论文中，不再需要来回剪贴了。不过要运行这个工具，必须在安装 MATLAB 时，把 Notebook 软件工具装入系统。

3.1.3 图形文件的转储

可以把 MATLAB 的图形文件转储为多种标准图形格式，以便用各种图形软件进行处理。存储时所用后缀可以是各种标准图形格式的后缀，如 gif、bmp、jpg 等。它们可由图形窗对图形进行存储而得到。在 MATLAB 5.1 及以前的版本中，图形的存储用 print 命令来完成。例如，先激活一个图形窗，然后在命令窗中键入：

```
print -dbitmap figaa
```

这就会在当前的目录下存入一个文件名为 figaa.bmp 的图形文件。要知道其他格式的命令，可以参阅 help print。

在 MATLAB 5.3 及 MATLAB 6.0 版本中，只要单击图形窗的菜单项【File】的子菜单【Export】（导出），就会出现图 3.1 的界面。在【Save as Type】中选定存储格式，给出文件名，再单击【Save】，即可完成图形的存储。这里用【Export】表示 MATLAB 把图形转储为其他软件的格式，是软件之间的接口转换。这样生成的文件，不属于 MATLAB 文件的范畴。

图形窗上还有一排图标组成的工具栏，可以完成图形编辑、缩放、旋转、加字符标注等功能，将鼠标移至该图标上时，会出现其功能的说明，此处无需一一介绍，读者可以在试

用中学习。

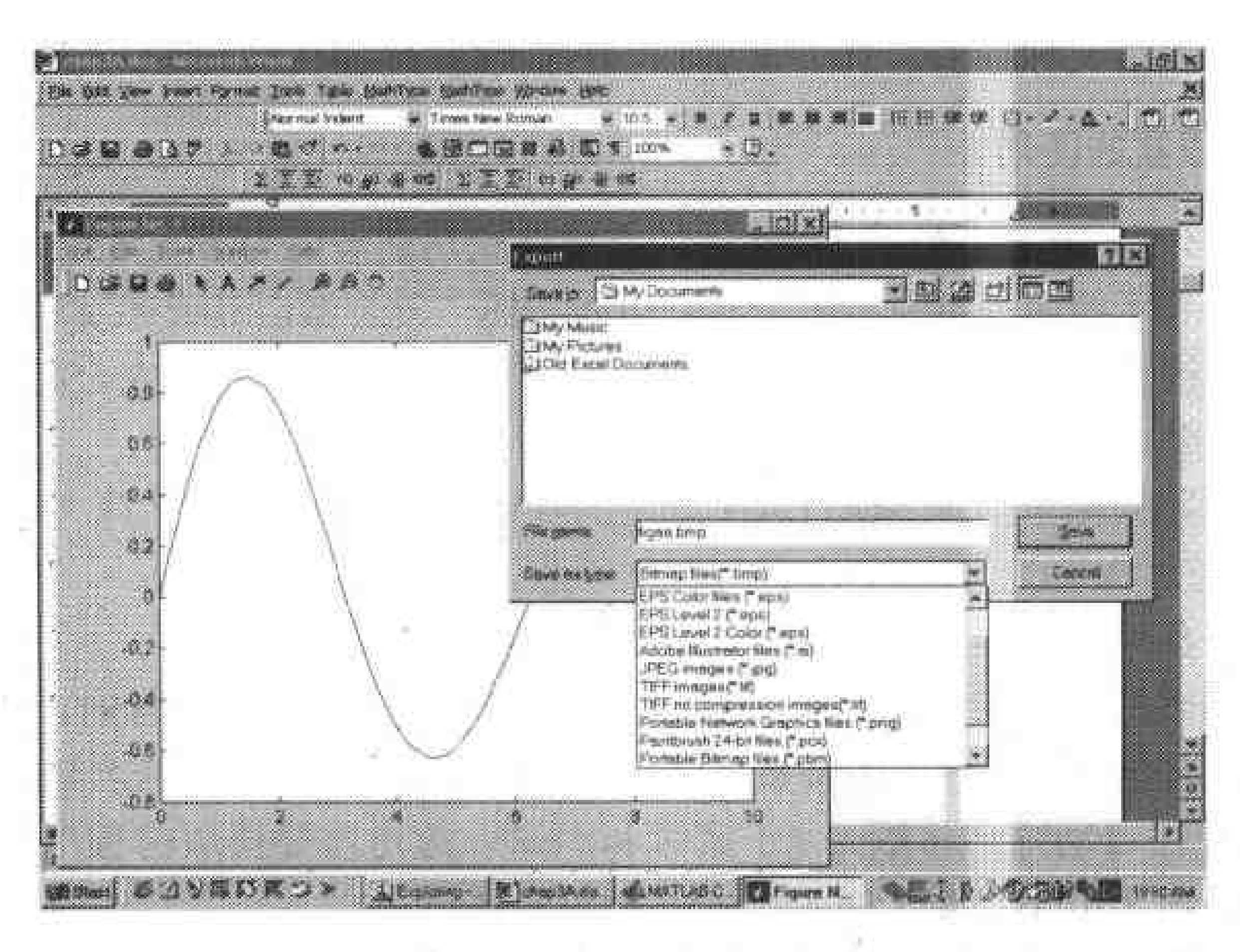


图 3.1　MATLAB5.3 及 MATLAB6.0 的图形窗及其转储（导出）界面

3.1.4　低层输入输出函数库

MATLAB 可以用 load 和 save 命令来保存和提取数据，其数据可以是 mat 或 ASCII 码格式，这已在前面讲过。但这只适合于 MATLAB 环境自身。作为一种科学计算软件，与其他软件系统进行直接的（没有人参与的）数据交换是十分重要的，它可以避免人为差错和运行低效。通过输入输出文件进行数据交换是有效的方法之一。因为几乎任何算法语言都有有限的几种输入输出文件格式（比如二进制格式和 ASCII 码字符格式），MATLAB 可以用这几种格式进行读写，也就保证了它可以在这一级上与其他语言相连接。例如将其他软件产生的或仪器测量的数据自动读入 MATLAB，再进行分析处理并绘成图形输出等。读不同格式的文件要用不同的命令，这个库中的命令见表 3.3。

表 3.3　低层文件输入输出库命令(iofun) (j)

文件开闭及 I/O	fopen	打开文件	fscanf	从文件读入格式化数据
	fclose	关闭文件	fprintf	把格式化数据写入文件
	fread	从文件读入二进制数据	fgetl	从文件读入一行，去掉新行字符
	fwrite	把二进制数据写入文件	fgets	从文件读入一行，保留新行字符
文件定位	ferror	询问文件 I/O 的出错状态	ftell	提取文件位置指针
	feof	测试文件结尾	frewind	倒回文件
	fseek	设置文件位置指针		
字符串及文件名处理	sprintf	把格式化数据写入字符串	sscanf	从字符串中读取格式化数据
	MATLABroot*	MATLAB 安装的根目录	partialpath*	部分路径名
	filesep	本平台的目录分割符	mexext	本平台的 MEX 文件名后缀
	pathsep	本平台的路径分割符	fullfile	从各部分构成全文件名
	tempdir	获取当前目录	tempname	获取当前文件

续表

文件输入输出	load	从 MAT 文件下载到工作空间	save	把工作空间变量存入 MAT 文件
	dlmread	从 ASCII 码分隔文件读取矩阵	dlmwrite	把矩阵写入 ASCII 码分隔数据文件
	wk1read	读 WK1 文件	wk1write	在 WK1 格式的文件中写入矩阵
图像声音 I/O	imread	从图形文件读出图像	imfinfo	返回图像文件的信息
	imwrite	把图像存入图形文件		
	wavwrite	写入 WAVE(".wav")声音文件	wavread	读出 WAVE(".wav")声音文件

如果要在一个二进制文件 aaa.bin 中写入工作空间中的变量 x，则其程序为如下两条语句：

```
fid1=fopen('aaa.bin','r+');  %打开 aaa.bin, 'r+'表示可读可写。fid1 为文件标识
N=fwrite(fid1,x,'float')     %将 x 以 float(浮点)格式写入 fid1 文件,返回实际写入的元素数 N
```

从数据文件读出变量是一个逆过程，例如，要从 aaa.bin 读入二进制数据并将它赋值给 A，程序可编写如下：

```
frewind(fid1)
fid1=fopen('aaa.bin','r+');       %打开 aaa.bin,'r+'表示可读可写. fid1 为文件标识
A=fread(fid1,[5,5],'float')
```

注意到这个程序比写入时多了第一行，因为文件的读写犹如磁带，写入以后必须倒带才能重放，要先键入倒带命令 frewind（fid1），而第 3 句表示将 fid1 文件中的前 25 个数据以 float（浮点）格式读出，列成 5×5 阶矩阵，赋予变量 A。如果以后还有从 fid1 文件读出的语句，就将从第 26 个数据开始。

输入输出的格式必须相同。MATLAB 内部本来只有一种双精度格式，现在要变换为其他语言中的多种数据类型，所以会很不适应。读者应在学了 C 语言或其他语言后再来理解本节。库中每个命令的具体用法可参看 help 文本，此处不多占篇幅。

在进行音频信号或图像处理时，需要与声音文件及图像文件接口。MATLAB 也提供了相应的命令。

在 MATLAB 中还有动态数据交换的函数库（dde）。利用它可以不经过“文件”这个中间环节而直接把运行 MATLAB 的计算机和运行其他软件的计算机之间通过网络进行数据交换。使 MATLAB 与其他软件平台之间的双向调用成为可能，这个函数库中的内容见表 3.4。

表 3.4　客户机函数库（dde）(g)

动态数据交换	ddeadv	建立链接	ddereq	从应用取得数据
	ddeexec	送出执行字符串	ddeterm	结束 DDE 对话
	ddeinit	DDE 对话初始化	ddeunadv	卸除链接
	ddepoke	把数据送到应用		

3.1.5　与 C 和 FORTRAN 子程序的动态链接

MATLAB 本身是用 C 语言编写的，它的丰富的科学计算子程序库中的许多经典部分来自久经考验的 FORTRAN 程序库。它可以直接调用经过一定的处理后的 C 和 FORTRAN 可执行文件，因而使执行这些子程序的速度与 C 语言及 FORTRAN 语言相同。这些可执行文

件就是后缀为 mex 的文件。除了 MATLAB 中已有的 mex 文件外，用户也可把自己找到的其他可执行文件加入系统。

MATLAB 高级工具箱中还有 C 编译器，可把 MATLAB 语言编写的子程序编译成 C 语言程序，以期提高它的运行速度。MATLAB 6.0 是用 Java 语言扩展的，因此也为它今后充分利用 Java 的功能创造了有利条件。

3.2　MATLAB 的文件管理系统

3.2.1　安装后的 MATLAB 文件管理系统

如果用光盘来安装 MATLAB 软件，不管版本有何差别，其过程和其他软件相仿，此处从简。安装后的 MATLAB 根目录（通常表为 MATLABroot）下，至少有 bin、extern、help、toolbox 这 4 个子目录，其中子目录 bin 包含了 MATLAB 所要用到的二进制文件。启动 MATLAB 的执行文件 matlab.exe 就在这个目录中，双击这个文件就可以启动 MATLAB 软件。子目录 extern 包含了 MATLAB 所要用到的外部文件；子目录 help 包含了 MATLAB 的各种帮助文件，如果有下一级子目录 pdf_doc，则其中将包括 MATLAB 及其工具箱的说明书，那是十分有用的资料。子目录 toolbox 包含了 MATLAB 的各种函数库及已装入的作为下一级子目录的工具箱名称等，它至少应有 local 和 matlab 两项，其中 matlab（注意用的是小写）又有 22 个子目录，分别是本书第 1～4 章介绍的 MATLAB 中的基本函数集，如 datafun、elfun 等。通常在 MATLAB 根目录下，还应该建立一个用户自己的子目录，例如 user 或 work，以便把用户自编的应用程序存在这个子目录下，免得与系统中原有的文件系统混淆。

3.2.2　MATLAB 自身的用户文件格式

MATLAB 的用户文件通常包括以下几类：

● **程序文件**　包括主程序和函数文件，其后缀为.m，即 M 文件。通常它由文本编辑器生成。MATLAB 的各个工具箱中的函数，大部分也是 M 文件。

● **数据文件**　其后缀为.mat。在 MATLAB 命令窗中，用 save 命令存储的变量，在默认条件下就生成这类文件。

● **MATLAB 的可执行文件**　其后缀为.mex，它们由 MATLAB 的编译器对 M 文件进行编译后生成。其运行速度远高于直接执行 M 文件的速度。

此外，用 Simulink 工具箱建模，会生成模型文件（后缀为.mdl）和仿真文件（后缀为.s），这些是 MATLAB 自身的文件格式。

3.2.3　文件管理和搜索路径

MATLAB 管理的文件范围由它的搜索路径来确定。该搜索路径由 MATLAB 启动文件 MATLABroot\toolbox\local\matlabrc.m 来规定。其中有一段程序列出了所有由它管理的文件目录名称（在 MATLAB 5.x 和 MATLAB 6.0 中，这段程序写成名为 pathdef.m 的子程序），这名称要列到最低层子目录。例如，MATLABroot\toolbox\matlab\elfun。当然，这些子目录不只限于 MATLAB 根目录下的范围，整个计算机资源管理器文件系统中的任何一个底层文

件夹，都可以列入 MATLAB 的搜索路径，在这些文件夹中的文件都可以被执行。反之，如果用户编写的程序未存入 MATLAB 搜索路径的子目录中，则 MATLAB 将找不到它，因而也无法运行这个程序。

要显示或修改搜索路径，可以用 path 命令。

- path　　列出 MATLAB 的搜索路径；
- path(path，'新增路径名')　　在原搜索路径群中加入一个新路径。

例如在 C 盘中已有一个子目录 user1，要把它放入 MATLAB 的搜索路径上，可键入：path(path,'c:\user1')。注意这个子目录必须先建立好，使用 path 命令才有效，否则 MATLAB 找不到这个子目录，它会显示“无此子目录，命令无效”，并拒绝执行。

用这个方法必须键入搜索路径，有时它会很长，费时又易出错。为了改进它，MATLAB5.x 中设置了修改搜索路径的控件。按下命令窗中工具栏右边第 2 个按钮，就出现如图 3.2 所示的对话框。

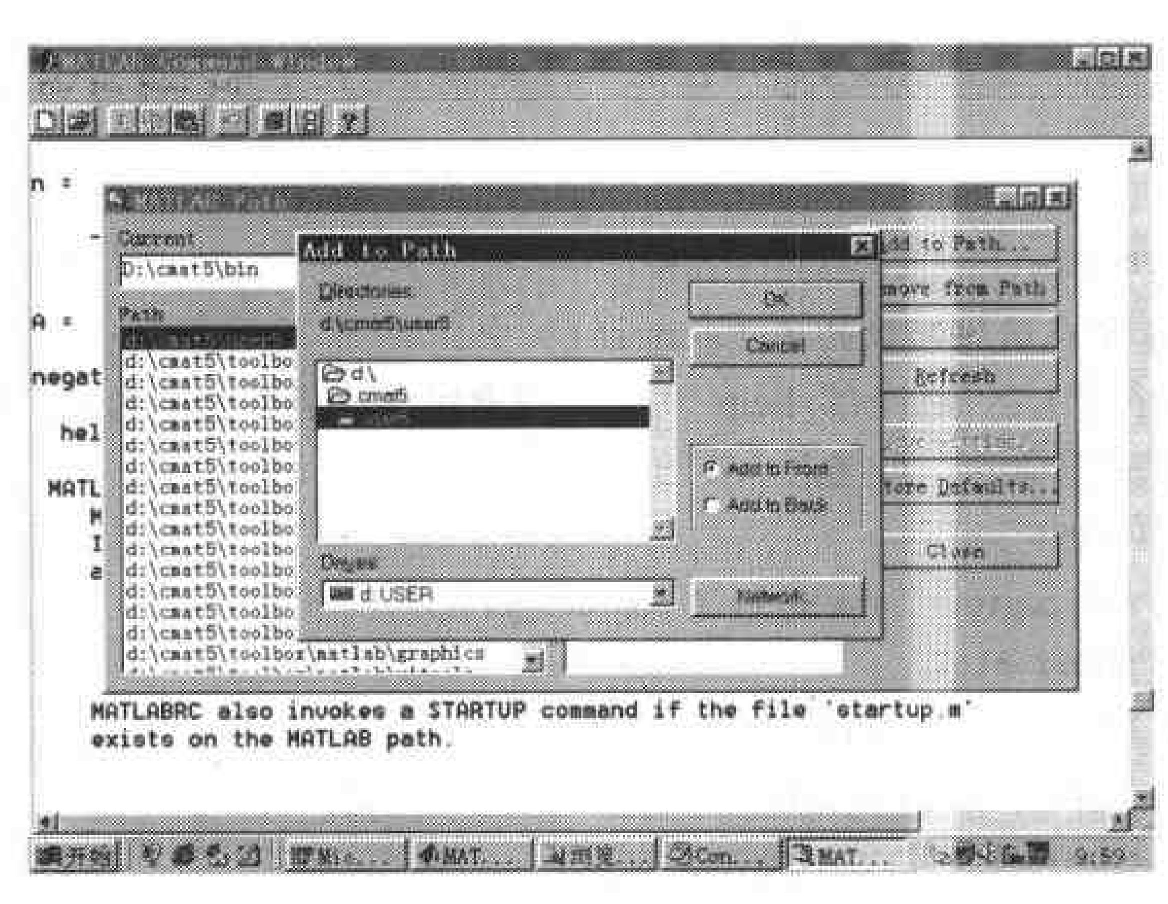


图 3.2　MATLAB5.x 中的修改搜索路径的对话框

图 3.2 中左方的大框显示出 MATLAB 已有的搜索路径，右方则是几个按钮。假如要把一个已经存在于系统中的子目录 user5 放入 MATLAB 的搜索路径下，可按如下步骤进行：

（1）用鼠标单击【Add to Path】按钮，会弹出一个系统文件搜索框，从中找到打算放入 MATLAB 搜索路径的文件夹 user5。将它选中，并单击【OK】按钮，此时该文件夹就被放入了 MATLAB 搜索路径。

（2）用鼠标单击右上角的图标，缩小或关闭搜索路径对话框。如果是关闭，系统会提问“是否要长期保存新加入的路径”，答复 Yes，则新路径就会一直放在 MATLAB 中；如果只是临时用一下这个子目录，以后不想把它长期保存在 MATLAB 搜索路径下，则不要关闭，只把它缩小。待运行完 MATLAB 的程序，打算关机时，再关闭搜索路径对话框。

MATLAB 6.x 的搜索路径设定对话框是靠主菜单上【File】的子菜单项【Set Path】激活的，其他做法大同小异，此处从略。

3.2.4　与目录和搜索有关的命令

- dir　　列出当前目录下的文件和子目录名；
- cd　　改变当前目录，如要往上改，用 cd ..；如要往下改，用 cd　[下一级子目录名]；
- delete　删除某个文件。

说明　这 3 个都是 DOS 操作系统的命令，在 MATLAB 中同样有效。

- what [子目录名]　　列出该子目录下的 MATLAB 自身的文件名，包括所有

后缀为 m 的 MATLAB 程序文本文件;
后缀为 mex 的 MATLAB 二进制执行文件;
后缀为 mat 的 MATLAB 的数据文件;
后缀为 mdl 的 MATLAB 的仿真模型文件;
……

- which [文件名]　　显示该文件所在的子目录路径。便于查看或修改它。例如键入：

```
which path
```

则显示

```
c:\matlab\toolbox\matlab\general\path.m
```

说明　path 命令在通用函数的库（general）中。

- lookfor [字符串]　　在全部 help 文件中搜索包含该字符串的内容。例如，想找到所有与等高线绘制有关的命令，可键入

```
lookfor contour
```

得

```
CLABEL    Add contour labels to a contour plot.
CONTOUR   Contour plot.
CONTOUR3  3-D contour plot.
CONTOURC  Contour computation.
MESHC     Combination MESH/CONTOUR plot.
SURFC     Combination SURF/CONTOUR plot.
```

3.2.5　搜索顺序

在 MATLAB 执行程序时，当遇到一个字符串时，如何判别该字符串的意义呢？它按如下的顺序（优先级）与已有的记录相比较：工作空间的变量名→内部固有变量名→mex 文件名→M 文件名。如果两个名字相同，它只认优先级高的名字。比如用户在工作空间中给 i 赋了值，那么系统就不会取内部固有变量中设定的虚数 i；如果用户在程序中设立了一个与 MATLAB 函数同名的变量，则每次调用此名字时，出现的将是用户自定的变量，调不出 MATLAB 中的函数。所以在自设变量名时要防止与 MATLAB 中的函数重名。

MATLAB 中也有函数同名只是后缀不同的情况。因为 mex 后缀是二进制的执行文件，它的运行速度比 M 文件快得多，所以会优先执行它。mex 文件通常是对 M 文件作编译后生成的。无法阅读也不好修改。

3.3　MATLAB 6.0 的开发环境

3.3.1　桌面系统的内容

第 1 章中初步介绍了 MATLAB 的 4 个基本视窗。其中的文本编辑器和文件管理器是借用 Windows 的现成工具。从本章的介绍可以看出，随着系统的升级，它们都不断升级。到 MATLAB 6.0 则发展到一个新阶段，它把多种开发工具集成为 MATLAB 桌面系统。该系统由桌面平台以及组件组成。包含如下 8 个组成部分：命令窗口（Command Window）、历史命令窗口（Command History）、资源目录本（Launch Pad）、当前路径浏览器（Current Directory Brower）、帮助浏览器（Help Browser）、工作空间浏览器（Workspace Browser）、数组编辑器（Array Editor）以及程序编辑调试器（Editor-Debugger）。它们的功能简述如下:

（1）命令窗口

第 2 章中的全部工作都是在命令窗中完成的，所以不必更多解释。

（2）历史命令窗口

历史命令窗口用于纪录并显示本次工作进程中曾键入的全部行命令。利用它可以方便地修改和输入较长的行命令，或把多个有用的行命令挑选出来，组成一个完整的程序文件。因此，这是一个很有用的工具。

（3）资源目录本

资源目录本用于把用户在当前系统中安装的所有 MATLAB 产品说明、演示以及帮助信息的目录集成起来，便于用户迅速调用查阅。

（4）当前路径浏览器

当前路径浏览器用于随时显示系统当前目录下的 MATLAB 文件信息，包括文件名、文件类型、最后修改时间以及该文件的说明信息等。

（5）帮助浏览器

所有的帮助信息都可以在该浏览器中显示。而且用户可以对原有的帮助信息编辑取舍，或加入自己的注解，形成自己的帮助文件。

（6）工作空间浏览器

工作空间浏览器用于显示所有目前保存在内存中的 MATLAB 变量的名称、数学结构、字节数以及类型，并与按下工作空间查看按钮或键入 whos 命令所得的结果相同。只是在工作空间浏览器中，还可以对变量进行编辑或图形操作。

（7）数组编辑器

用户可以直接在数组编辑器中修改所打开的数据，甚至可以更改该数据的数学结构以及显示方式。

（8）程序编辑调试器

以上 8 项组件中，许多是 MATLAB 5.x 就有的。但其中有些部分的功能较低，例如只能显示，不能编辑修改。MATLAB 6.0 把它们集成起来，构成桌面系统。而且各组件都独立地构成视窗，具有自己的菜单和工具条，可以对视窗中的内容进行编辑和存储，这就使它们的功能更强大，使用更方便。

3.3.2　桌面命令菜单简介

图 1.3 的第一行给出了 MATLAB 6.0 的桌面命令菜单区，它包括【File】、【Edit】、【View】、【Web】、【Window】、【Help】等 6 项。在第 6 项的右边，增加了一个显示当前目录的信息区。在主菜单上增加了 Web 项，表明它在联网功能上的加强，它的其他功能扩展主要反映在子菜单中。

在【File】下的子菜单中，增加了【Import Data...】(数据导入)、【Save Workspace As...】（将工作空间保存为文件）、【Set Path...】（搜索路径设定）、【Preference...】（选择）等选项。

在【Edit】下的子菜单各项，与一般文本编辑命令相仿，此处从简。只有【Paste Special】选项有些特别。用它可打开数据输入向导，将剪贴板的数据输入到 MATLAB 工作空间中。

在【View】下的子菜单中，增加了【Desktop Layout】（桌面布局）、【Undock Command Window】（与命令窗分离）、【Command Window】、【Command History】、【Current Directory】、【Workspace】、【Launch Pad】、【Help】、【Current Directory Filter】以及【Workspace View Options】等选项；用以选定观察的视窗。

在子菜单项【Desktop Layout】之下又有下一级子菜单，利用它可以同时显示两个以上的视窗。显示方案列在下一级子菜单中，分别为【Default】（默认方式，同时显示【Commend Window】、【Launch Pad】和【Commend History】3 个视窗）、【Command Window Only】、【Simple】（同时显示【Commend Window】和【Commend Histroy】或【Workspace】2 个视窗）、【Short History】、【Tall History】（各显示【Commend Window】、【Current Directory】和【Commend History】3 个视窗，但排列不同），以及【Five Panel】（同时显示 5 个视窗）等。

【Web】是 MATLAB 6.0 新增的菜单项，通过该菜单项可以直接得到 MATLAB 的网络资源，当然此时系统必须在联网状态。它提供的网址如下。

（1）The MathWorks Web Site　　将连接 Mathworks 公司的主页：
http://www.mathworks.com/；

（2）Technical Support Knowledge Base　　将连接 Mathworks 公司的技术支持网页：
http://www.mathworks.com/support；

（3）Products　　将连接产品介绍页面：http://www.mathworks.com/products；

（4）Membership　　将介绍 Mathworks 公司的会员制度：
http://www.mathworks.com/membership。

【Window】菜单项提供了在已打开的各 MATLAB 视窗之间的切换功能，也可以用它关闭全部视窗。

【Help】下的子菜单项【Full Product Family Help】、【MATLAB Help】、【Using the Desktop】、【Using the Command Windows】和【Demo】分类提供了进入各类帮助信息系统的入口。

3.3.3　MATLAB 6.0 的用户界面

在 Windows 的资源管理器中，双击 MATLAB\bin\matlab.exe 文件，或者双击已经处于 Windows 桌面上的 MATLAB 图标，就可以启动 MATLAB 软件环境。它首先显示出一个标

志 MATLAB 6.0 的图形标志，如图 3.3 所示。如果系统的【View】下的【Desktop Layout】处于默认（Default）状态，则经过几秒钟后，屏幕上将出现图 3.4 所示的 MATLAB 桌面平台，它由命令窗口（Command Window）、历史命令窗口及资源目录本 3 个视窗组成。根据用户的需要，可以选定并激活相应的视窗进行操作。例如，在上述界面中关闭后两个视窗，或在【Desktop Layout】中选定【Command Window Only】，就会出现图 1.3 所示的单一命令窗。

图 3.4 中的图形窗是由命令窗中的 plot 语句打开的。画出这个衰减正弦曲线的有效命令有 3 条，要想把这 3 条命令组织起来，形成一个程序文件，就可以利用历史命令窗。如图 3.4 所示，在历史命令窗中用【Ctrl】+鼠标单击，一次选出这 3 条语句，进行【Copy】，并将他们【Paste】到文本编辑器中去，再【Save】起来即可。

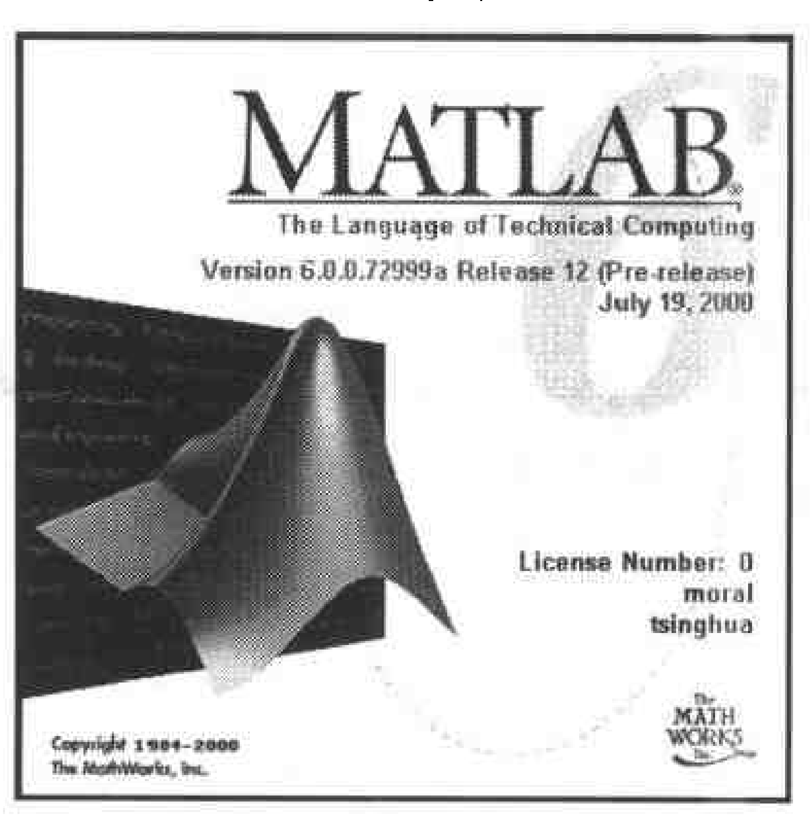


图 3.3　MATLAB 6.0 的标志界面

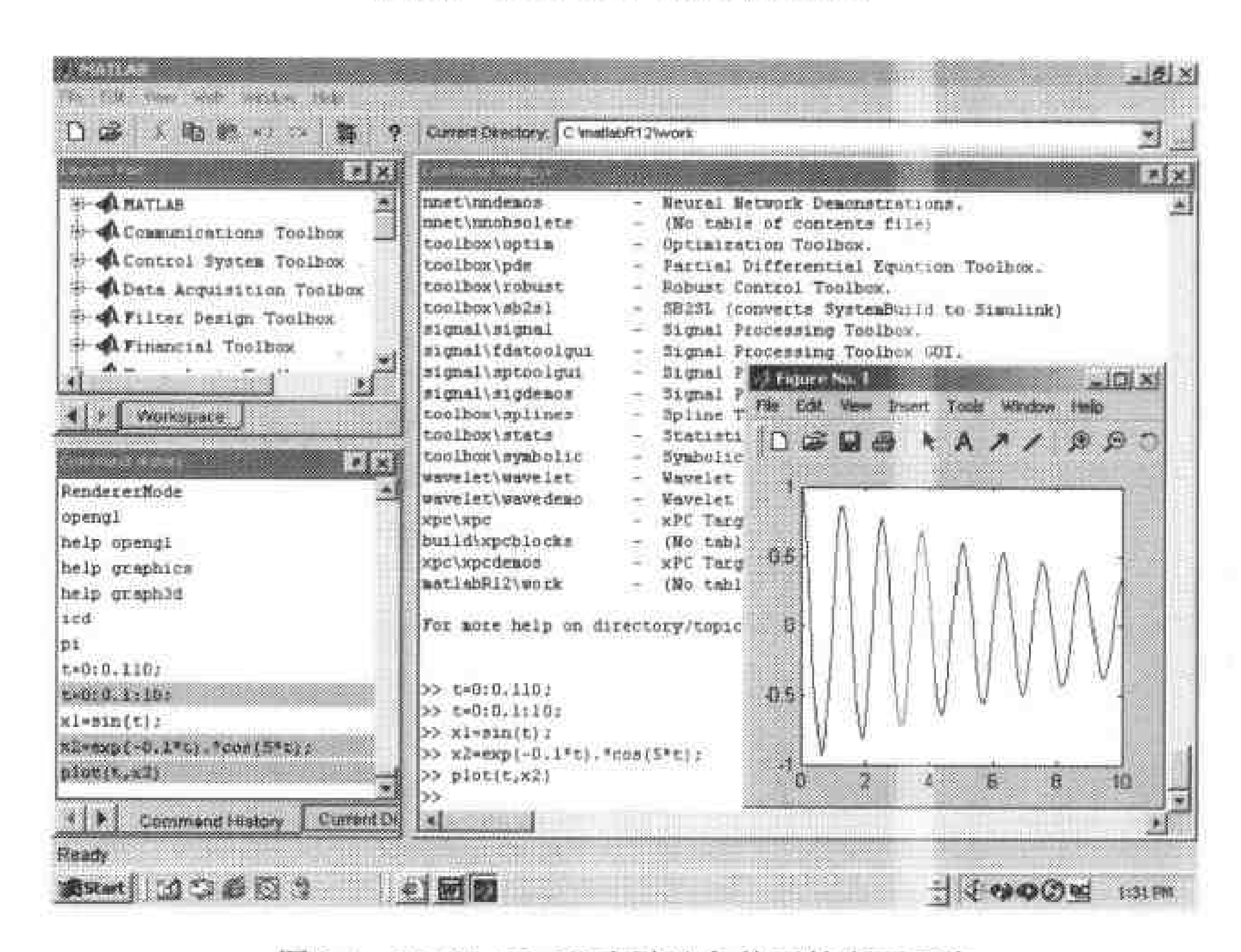


图 3.4　MATLAB 6.0 在默认条件下的桌面平台

再来看看 MATLAB 6.0 的帮助浏览器，在【View】下选择【Help】项，就会出现图 3.5 所示的帮助浏览器，左边是目录栏，右边是帮助的内容。在这个图上可以看到，目录中有三个 angle 命令，它们出现在不同的工具箱中，图 3.5 中显示的是第一个 angle 命令的意义和用法。另外，浏览器的目录部分还给出了多种搜索与查找方法。

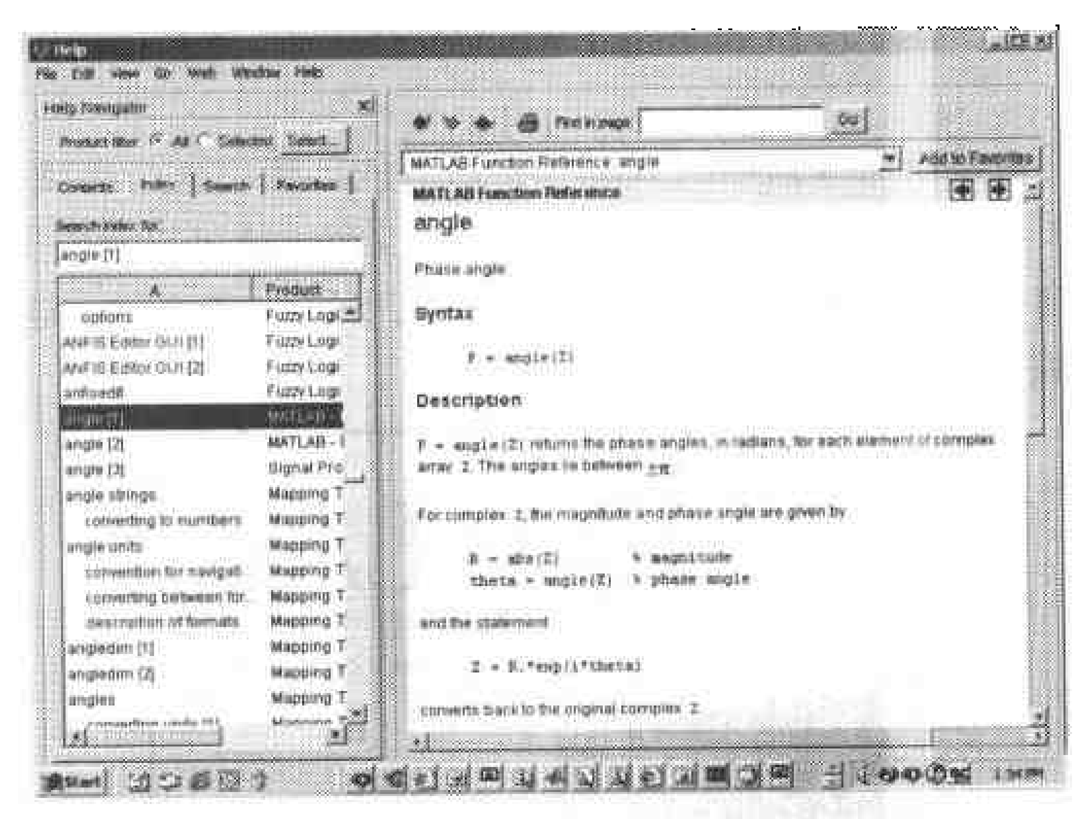


图 3.5　MATLAB 6.0 帮助浏览器

第 4 章　MATLAB 的其他函数库

在前 3 章中，已介绍了 MATLAB 基本部分的函数库中最常见的十几个，这些内容实际上已远远超过了 FORTRAN 语言和 C 语言在科学计算上的性能，对于大学低年级的初学者来说，已经足以应付各种大学课程计算题的需要。余下的这几个函数库中，有些是涉及深一些的数学和物理概念，对大学低年级读者而言有些超前，不易很快接受，可以留到高年级再深入；有些则是用来编写较高级的程序用的，要对 MATLAB 编程比较熟悉后再学。单独把这些内容放在一章中，可以避免对初学者造成拦路虎。这一章内容具有独立性，其各节也有独立性，读者可以跳过整章或整节，向后阅读。但要读懂本章各节，必须以前 3 章为基础，并具备相应的数学和理论知识。

4.1　数据分析函数库(datafun 函数库)

4.1.1　基本的数据分析

MATLAB 的基本数据处理功能是按列向进行的，因此要求待处理的数据矩阵按列向分类，而行向则表示数据的不同样本。例如，10 个学生的身高及 3 门课程分数列表如下：

```
data  =  154     49     83     67
         158     99     81     75
         155    100     68     86
         145     63     75     96
         145     63     75     96
         141     55     65     75
         155     56     64     85
         147     89     87     77
         147     96     54    100
         145     60     76     67
```

进行简单数据处理的命令见表 4.1。

表 4.1　一些数据处理命令的结果

命　令	功　能	身　高	课　程　1	课　程　2	课　程　3
max(data)	求各列最大值	158	100	87	100
min(data)	求各列最小值	141	49	54	67
mean(data)	求各列平均值	149.2	73.0	72.8	82.4
std(data)	求各列标准差	5.7504	20.4070	10.0421	12.0757
median(data)	求各列中间元素	147	63	75	81
sum(data)	求各列元素和	1492	730	728	824
trapz(data)	梯形法求积分	1342.5	675.5	648.5	757.0

其中大部分命令的意义很明确，不需解释。

- std（标准差）　是指列中 N 个元素与该列平均值 mean(data)之差的平方和开方。即

$$\mathrm{std(data)}=\sqrt{\sum_{N}\left(\mathrm{data}-\mathrm{mean(data)}\right)^{2}}$$

- trapz（求积分）　可以看成求和，梯形法求积分近似于求元素和，其差别在于梯形法是把相邻两点数据的平均值作为数据点。10 个数据只能产生 9 个数据点，相加以后比元素和少一组数据，把它乘以步长才真正表示了面积。其差额为半个首点和半个末点的数据和，即

$$\mathrm{trapz(data)}=\mathrm{sum(data)}-0.5(\mathrm{data(1)}+\mathrm{data(N)})$$

在 N 很大时两者的误差很小。

有些数据处理命令的结果不是一个标量而是一个列向量，为了节省篇幅，只取数据中的前 3 行，其结果见表 4.2。注意结果一般与原数据具有同样的行数，只有求差分（diff）会减少一行，因为它是求相邻行之间的差。另外，cumtrapz 是 MATLAB 5.x 中的新增函数，相当于累计求面积。

表 4.2　产生列结果的数据处理命令

命　令	功　能	身　高	课 程 1	课 程 2	课 程 3
cumsum(data(1:3,:))	列向累加和	154	49	83	67
		312	148	164	142
		467	248	232	228
cumprod(data(1:3,:))	列向累乘积	154	49	83	67
		24332	4851	6723	5025
		3771460	485100	457164	432150
diff(data(1:4,:))	列向差分	4	50	-2	8
		-3	1	-13	11
		-10	-37	7	10
sort(data(1:3,:))	列向重新排序	154	49	68	67
		155	99	81	75
		158	100	83	86
cumtrapz(data(1:4,:))*	列向累加积分(相当于不定积分)	156.0000	74.0000	82.0000	71.0000
		312.5000	173.5000	156.5000	151.5000
		462.5000	255.0000	228.0000	242.5000

4.1.2　用于场论的数据分析函数

以下的几个命令是用于场论的。

- gradient　用来求二维场和三维场的近似梯度，例如根据电位分布求电场就可用这个函数。
- del2　是二维场和三维场的拉普拉斯算子。
- cross　为两个向量的矢量积。
- dot　为两个向量的数量积。

设 $\boldsymbol{i}$，$\boldsymbol{j}$，$\boldsymbol{k}$ 为沿 x，y，z 方向的单位向量，则对于两向量 $\boldsymbol{a}=a_x\boldsymbol{i}+a_y\boldsymbol{j}+a_z\boldsymbol{k}$ 和 $\boldsymbol{b}=b_x\boldsymbol{i}+b_y\boldsymbol{j}+b_z\boldsymbol{k}$。

向量的矢量积为（叉乘）： $\boldsymbol{a}\times\boldsymbol{b}=(a_yb_z-a_zb_y)\boldsymbol{i}+(a_zb_x-a_xb_z)\boldsymbol{j}+(a_xb_y-a_yb_x)\boldsymbol{k}$

向量的数量积为（点乘）： $\boldsymbol{a}\cdot\boldsymbol{b}=a_xb_x+a_yb_y+a_zb_z$

在 MATLAB 中这两个向量可表为：

```
a=[ax, ay, az]; b=[bx, by, bz];
cross(a, b) = [ay*bz-az*by, az*bx-ax*bz, ax*by-ay*bx];
dot(a, b) = a*b';
```

4.1.3 用于随机数据分析的函数

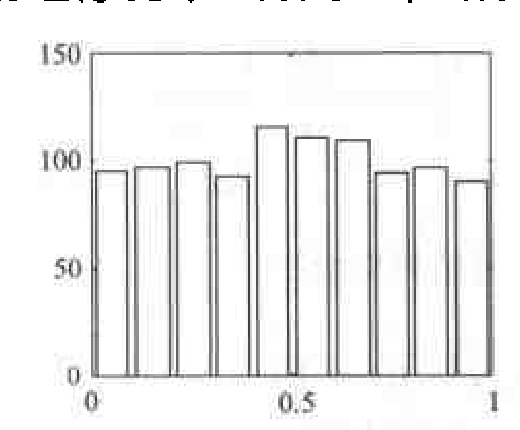
MATLAB 有两个产生随机数的命令。

➢ rand(m,n)　产生在 0 与 1 之间均匀分布的 m 行 n 列随机数矩阵，其均值为 0.5，标准差（或均方根差）为 0.2887；

➢ randn(m,n)　产生正态分布的 m 行 n 列随机数矩阵，其均值为 0，标准差为 1；其分布情况可用直方图命令 hist(x,N)来显示。其中 N 表示直方图横坐标的分割数，其默认值为 10。例如：

```
x=rand(1,1000);hist(x)
y=randn(1,1000);
```

得出的两组图形如图 4.1 所示。hist(x)是把 1000 个 x 中，处于 0～0.1，0.1～0.2，…，0.9～1.0 各个区域中的数目分别清点出来，画成直方图。如果 x 真是均匀分布的，那这个图应该是水平直线，实际上，随机数规律是按统计方法确定的，所以各区域的数量一般参差不齐，只有数据量无限增加时，此图才无限接近于理想情况。hist（y，50）则把 y 的最小值和最大值之间等分成 50 份进行统计，得到一个钟形的，即正态分布的曲线。



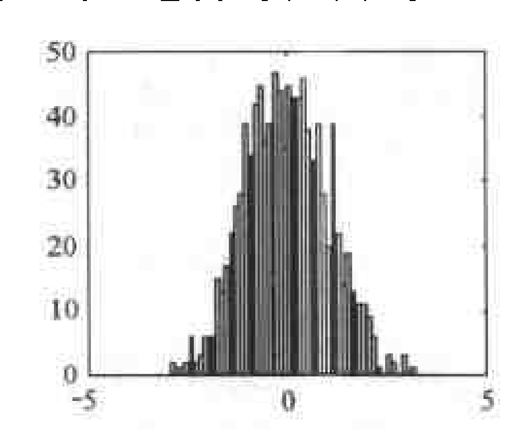


图 4.1 均匀分布与正态分布随机数直方图

4.1.4 用于相关分析和傅立叶分析的函数

相关分析（包括卷积）和傅立叶分析分别用于信号的时域和频域处理。这里给出了十几个函数，它们是整个信号处理计算的基础。

➢ corrcoef　给出两个同长信号的相关系数，例如对前面两个随机序列，键入

```
R=corrcoef(x, y)
```

得 `R = [1  0.0508  -0.0508  1]`

对角线上是 x 和 y 的自相关系数，这说明 x 和 y 的自相关性很强，而互相关性则很弱。

➢ cov(x,y)　给出 x，y 的协方差矩阵，对上述 x，y，有

```
cov(x,y) = [  0.0785  -0.0148;  -0.0148  1.0782 ]
```

其主对角线上的值分别为 x 和 y 的均方差，即标准差的平方（因为是随机数，它不会严格等于理论值）。

➢ conv(x，y)　给出 x，y 的卷积。如果 x 是输入信号，y 是线性系统的脉冲过渡函数，则 x，y 的卷积给出系统的输出信号。卷积函数也用于多项式相乘，见 4.3 节。

➢ filter(b，a，x)　是根据输入信号 x 和线性系统特性求输出信号的函数。其不同在于系统的特性是以传递函数的分子多项式系数向量 b 和分母多项式系数向量 a 给出，而不是以脉冲过渡函数的形式给出。

➢ X = fft(x，N)　求出时域信号 x 的离散傅立叶变换 X。N 为规定的点数。N 的默认值为所给 x 的长度。当 N 取 2 的整数幂时变换的速度最快。通常取大于又最靠近 x 的幂次，即令 N=2 ^ nextpow2(length(x))。例如 x 的长度为 12，nextpow2(12)=4，N=2^4=16。多出的各点补以零。一般情况下，fft 求出的函数为复数，可用 abs 及 angle 分别求其幅度和相位。在画频谱图时往往最关心其幅频特性。

【例 4.1】 给出一个信号：

```
t=0:.001:3; u=sin(300*t)+2*cos(200*t) ;
```

它的幅频特性可用下列语句求得：

```
U=fft(u);plot(abs(U))
```

得出的频谱曲线如图 4.2（a)，它对采样频率呈对称形式。为了把它看得更清楚，把坐标缩小，键入 axis([0,300,0,3000])，得出的频谱曲线如图 4.2（b)。

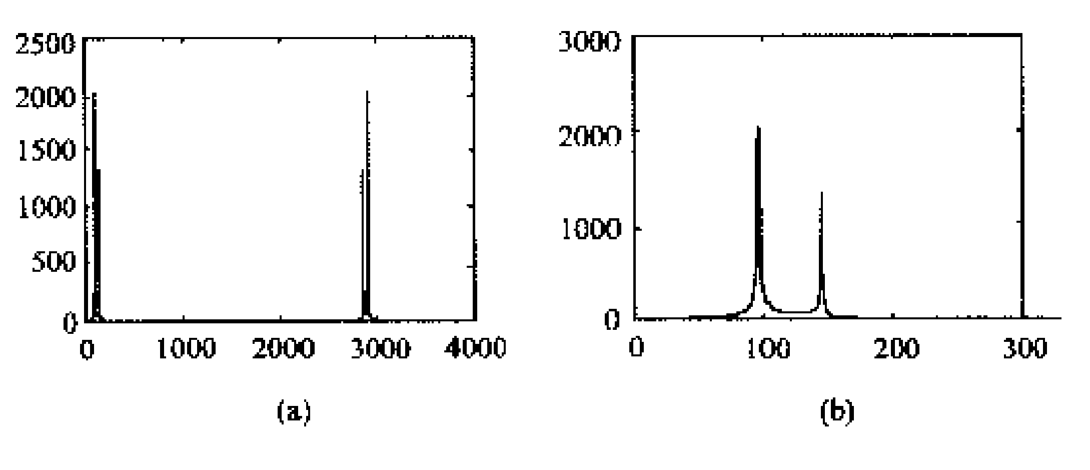


图 4.2 例题中信号的幅频曲线

➢ x=ifft(X)　傅立叶反变换函数，其用法与 fft 相仿。

➢ sound(u，s)　会在音箱中产生 u 所对应的声音，s 规定重放的速度，其缺省值为 8192（bps-bit per second)，见表 4.3。

表 4.3 数据分析和傅立叶变换函数库(datafun)(d)

基本运算	max	最大元素	sum	元素之和
	min	最小元素	prod	元素之积
	mean	平均值	cumsum	元素的累加和
	median	中间值	cumprod	元素的累积
	std	标准差	hist	直方图
	sort	按升序排列	trapz	用梯形法作定积分
	sortrows*	按升序排列行	cumtrapz*	梯形法作不定积分
差分	diff	差分函数和近似微分	gradient	近似梯度
	del2	五点离散拉普拉斯算子		

续表

相关运算	corrcoef	相关系数		
	cov	协方差矩阵		
滤波和卷积	filter	一维数字滤波	filter2	二维数字滤波
	conv	卷积和多项式相乘	conv2	二维卷积
	convn*	*n* 维卷积	deconv	反卷积和多项式相除
傅立叶变换	fft	离散傅立叶变换	ifft	离散傅立叶反变换
	fft2	二维离散傅立叶变换	ifft2	二维离散傅立叶反变换
	fftn*	*n* 维离散傅立叶变换	ifftn*	*n* 维离散傅立叶反变换
	fftshift	将零迟延移到频谱中心		
声音函数	sound	把向量放成声音	mu2lin	把 mu 规律编码变为线性信号
	soundsc	自动设比把向量放成声音	lin2mu	把线性信号变为 mu 规律编码

4.2 矩阵的分解与变换(matfun 函数库)

4.2.1 线性方程组的系数矩阵

在 2.2 节中提到了可以用矩阵除法来解线性方程组，本节将讨论有关解线性方程组的一些深入的问题及其工具函数。这些函数见表 4.4 中的矩阵分析和线性方程部分。

det(a)用以求方阵 a 的行列式。若 det(a)不等于零，则 a 的逆阵 inv(a)存在。线性方程组的系数矩阵只有满足这个条件，它的解才存在。rank(a)用以求任意矩阵 a 的秩。也就是它所能划分出的行列式不为零的最大方阵的边长。trace(a)求出矩阵主对角线上元素的和（即迹）。

如果 det(a)虽不等于零，但数值很小，接近于零，则这样的线性方程组称为病态的。其解的精度比较低。为了评价线性方程组系数矩阵的病态程度，用了条件数（condition number）的概念。条件数愈大，方程病态愈重，解的精度愈低。若系数矩阵的条件数很大而又拿它作除数（即解此线性方程组）时，MATLAB 会提出警告：“条件数太大，结果可能不准确”，求条件数的函数为 cond(a)，a 可以不是方阵。

在线性方程组 A*x=B 中，系数矩阵 A 的行数 n 表示方程的数目，其列数 m 表示未知数的数目。在正常情况下，方程个数等于未知数个数，即 n=m。A 为方阵，A\B 意味着 inv(A)*B。实际上，对于方程个数大于未知数个数（n>m）的超定方程组，以及方程个数小于未知数个数（n<m）的不定方程组，MATLAB 中 A\B 的算式都仍然合法。前者是最小二乘解；而后者则是令 x 中 m-n 个元素为零的一个特殊解。这两种情况下，因为 A 不是方阵，其逆 inv(A)不存在，解的 MATLAB 算式均为 x=inv(A'*A)*(A'*B)。把 pinv(A)=inv(A'*A)*A'定义为伪逆函数，则 A\B 就等于 pinv(A)*B。由于 MATLAB 中引入了伪逆的概念，除矩阵 A 就可以不是方阵。

【例 4.2】 求下列矩阵的行列式、秩、逆阵、迹和条件数。

```
a =  2    9    0    0
     0    4    1    4
     7    5    5    1
     7    8    7    4
```

解：■ MATLAB 实现

```
    det(a) =  -275
rank(a) =    4
inv(a) = -0.0727      0.4255      0.7855      -0.6218
          0.1273     -0.0945     -0.1745       0.1382
          0.0000     -0.6000     -0.8000       0.8000
         -0.1273      0.4945      0.3745      -0.3382
trace(a) =  15
cond(a) =  33.4763
```

4.2.2　矩阵的分解

矩阵可以分解为几个具有特殊构造性质的矩阵的乘积，这是分析矩阵的一种重要手段，而这种分解在数学计算上通常又是非常繁琐的工作。MATLAB 提供了一些现成的函数可供调用，主要有表 4.4 所列的几种。

表 4.4　矩阵函数和数值线性代数函数库（matfun）(m)

矩阵分析	norm	矩阵或向量的范数	null	零空间正交基
	normest	矩阵 2 范数的估值	orth	正交化
	rank	矩阵的秩	rref	缩减行梯次格式
	det	行列式(必须是方阵)	subspace	两个子空间之间的夹角
	trace	主对角线上元素的和		
线性方程	\ 和 /	线性方程求解	qr	正交三角分解
	chol	Cholesky 分解	cholinc*	不完全 Cholesky 分解
	cond	矩阵条件数	condest	1 范数条件数的估值
	rcond	linpack 逆条件数计算	nnls	非负最小二乘
	lu	高斯消去法系数矩阵	pinv	矩阵伪逆
	inv	矩阵求逆(必须是方阵)	lscov	协方差已知的最小二乘
特征值和奇异值	eig	特征值和特征向量	eigs	若干特征值
	poly	特征多项式(必须是方阵)	condeig	对应于特征值的条件数
	polyeig	多项式特征值问题	schur	Schur 分解
	hess	Hessenberg 形式	balance	均衡（改善条件数）
	qz	广义特征值	svd	奇异值分解
矩阵函数	expm	矩阵指数	expm2	用泰勒级数求矩阵指数
	expm1	用 M 文件求矩阵指数	expm3	用特征值求矩阵指数
	logm	矩阵对数	funm	通用矩阵函数的计算
	sqrtm	矩阵开方		
分解工具	qrdelete	从 QR 分解中删去一列	rsf2csf	实对角阵变为复对角阵
	qrinsert	在 QR 分解中插入一列	cdf2rdf	复对角阵变为实对角阵
	planerot	Given's 平面旋转		

● 三角分解（lu 分解）

它把一个任意方阵分解为一个准下三角方阵和一个上三角方阵的乘积。由于此函数有两个输出矩阵，其左端应有两个变量 l 和 u。

键入

```
[l,u]=lu(a)
```

得
```
l = 0.2857    1.0000    0         0
    0         0.5283    0.6838    1.0000
    1.0000    0         0         0
    1.0000    0.3962    1.0000    0
u = 7.0000    5.0000    5.0000    1.0000
    0         7.5714   -1.4286   -0.2857
    0         0         2.5660    3.1132
    0         0         0         2.0221
```

所谓准下三角阵 l 是因为必须交换两行才能成为真的下三角阵。它的行列式绝对值等于 1（可正可负），上三角阵 u 的行列式等于 a 的行列式。lu 分解只能对方阵进行，它常用于高斯消去法。

● 正交分解（qr 分解）

qr(a)把任意矩阵 n×m 阶 a 分解为一个正交方阵 q 和一个与 a 有同样阶数的上三角矩阵 r 的乘积。该方阵 q 的边长为矩阵 a 的 n 和 m 中之小者，且其行列式的值为 1。

【例 4.3】 求下列 3×5 阶矩阵 b 的 qr 分解。

```
b = 0.2190    0.6793    0.5194    0.0535    0.0077
    0.0470    0.9347    0.8310    0.5297    0.3834
    0.6789    0.3835    0.0346    0.6711    0.0668
```

解：■ MATLAB 实现

键入 `[q,r]=qr(b)`

得
```
q = -0.3063   -0.4667   -0.8297
    -0.0658   -0.8591    0.5076
    -0.9497    0.2101    0.2324
r = -0.7149   -0.6338   -0.2466   -0.6886   -0.0911
     0        -1.0395   -0.9490   -0.3390   -0.3189
     0         0        -0.0011    0.3805    0.2038
```

● 奇异值分解（svd 分解）

svd(a)把任意 n×m 阶矩阵 a 分解为三个矩阵的乘积，即 a=usv。其中 u，v 为 n×n 阶和 m×m 阶正交方阵，s 则为 n×m 阶的对角阵。对角线上的元素就是矩阵 a 的奇异值，其长度为 n 和 m 中的较小者。

例如对上述 b 作奇异值分解，键入

```
[u,s,v]=svd(b)
```

得
```
u = -0.4623    0.2273    0.8571
    -0.7822    0.3507   -0.5149
    -0.4176   -0.9085    0.0157
s = 1.7539    0         0         0         0
```

```
         0         0.7995    0          0          0
         0         0         0.3534     0          0
v =      0.2403   -0.6885    0.4927    -0.4748     0
         0.6872    0.1674    0.3027     0.4193     0.4819
         0.5158    0.4729    0.0506    -0.3723    -0.6076
         0.4102   -0.5151   -0.6122     0.3194    -0.2994
         0.1890    0.0944   -0.5369    -0.5985     0.5558
```

矩阵最大奇异值和最小奇异值之比就是它的条件数。即：

```
cond(b)=max(diag(s))/min(diag(s))
```

4.2.3　矩阵的特征值分析

eig(a)用来求方阵 a 的特征根和特征向量，其输出量有两个，即特征向量 e 和特征根 r。

键入　[e,r]=eig(a)

```
得  e =  -0.2568   -0.3834 + 0.4681i   -0.3834 - 0.4681i    0.6167
         -0.3481   -0.2177 - 0.2869i   -0.2177 + 0.2869i   -0.1850
         -0.4682    0.5152 + 0.2228i    0.5152 - 0.2228i   -0.6624
         -0.7705    0.4217 - 0.1060i    0.4217 + 0.1060i    0.3829
    r =  14.2004    0                   0                   0
          0         0.7495 + 5.2088i    0                   0
          0         0                   0.7495 - 5.2088i    0
          0         0                   0                  -0.6993
```

特征根是特征方程的根，矩阵的特征方程系数可用 poly 函数求出，再用 roots 命令也可求出其特征根。例如

键入　p=poly(a)

```
得  p =    1.0000  -15.0000   38.0000 -359.0000 -275.0000
而  roots(p) =   14.2004
                  0.7495 + 5.2088i
                  0.7495 - 5.2088i
                 -0.6993
```

结果与 eig 函数求出的特征根相同，但 roots 函数不能求特征向量。

4.2.4　特殊矩阵库(specmat)

有一些特殊构造的矩阵在矩阵变换中很有用处，MATLAB 4.x 将它们组成一个专门的函数库。读者在应用中遇到有关的矩阵，即可在此调用。其调用的参数和调用方法均可从 help 文本中查得，此处不多占篇幅。MATLAB 5.x 已把这个库并入 elmat 库中，见表 **2.1** 中的“特殊矩阵”栏。

4.3　多项式函数库（polyfun）

一元高次代数多项式 $a(x)=a_1x^n+a_2x^{n-1}+\cdots+a_nx+a_{n+1}$，在 MATLAB 中可以用它的系数向量 a=[a(1)，a(2)，…，a(n)，a(n+1)]来表示。其幂次已隐含在系数元素离向量右端的元素的间隔中。要注意，如果 x 的某次幂的系数为零，这个零必须列入系数向量中。在微分方程中，通过微分积分运算可把线性系统的特性表示为两个算子 s 的多项式之比。因此，多项式在近代信息和控制理论中有着十分重要的地位。

4.3.1　多项式的四则运算

【例 4.4】 设有两个多项式 $a(x)=2x^3+4x^2+6x+8$ 及 $b(x)=3x^2+6x+9$，要求对此两个多项式作如下运算。

● 多项式相乘(conv)　　conv 函数本来是卷积（convolution）的意思。但它也符合多项式相乘的运算规则。分析如下：

可以想象把系数向量 a 正常排列，而把 b 反转，先将 a(1)与 b(1)对齐：

```
a(1)     a(2)     a(3)     a(4)
b(3)     b(2)     b(1)
```

把上下对应的项相乘，a(1)*b(1)得出多项式乘积 c 的最高次项系数 c(1)；

将 b 右移一位，把上下对应的项相乘并求和，a(1)*b(2)+a(2)*b(1)为次高次项系数 c(2)；依此类推，可得到乘积 c 的全部系数。这种运算和卷积运算的规则完全相同。

■ MATLAB 实现

键入　a=[2,4,6,8],b=[3,6,9], c=conv(a,b)，得

```
a =    2    4    6    8
b =    3    6    9
c =    6   24   60   96   102   72
```

● 多项式相加　　MATLAB 规定，只有长度相同的向量才能相加。因此，必须把短的向量前面补以若干个零元素，才能用 MATLAB 的矩阵加法运算符。

■ MATLAB 实现

键入　d=a+[0,b]

得　d =   2   7   12   17

这种手工数两个多项式的长度再补零的方法是不可取的，必须要让计算机自动完成。为此可编一个子程序 polyadd.m，其内容为：

```
    function y=polyadd(x1,x2)
    n1=length(x1); n2= length(x2);
    if n1>n2 x2=[zeros(1,n1-n2),x2];
    elseif n1<n2 x1=[zeros(1,n2-n1),x1];
end, y = x1+x2;
```

这样，多项式相加就可写成：c = polyadd(a,b)；相减可另编一个子程序，或在 polyadd 的输入变元中加负号来实现。

● 多项式相除　相除是相乘的逆运算，但除法不一定除得尽，会有余子式。

■ MATLAB 实现

键入　[q, r]=deconv(c, a)

```
q =    3    6    9
r =    0    0    0    0    0    0
```

其中 q 是商式，r 是余子式。因为用的是相乘的数据 a 和 c，恰好除净。如令 a1=a+1，则有

```
a1 =    3    5    7    9
[q1, r1]=deconv(c, a1)
q1 =    2.0000    4.6667    7.5556
r1 =    0    0    0    7.5556    7.1111    4.0000
```

可以用商式与除式相乘，再加上余式的方法来检验:

```
c1=conv(q1, a1)+r1
```

得　c1 =   6   24   60   96   102   72

与 c 相同。

4.3.2　多项式求导、求根和求值

● 多项式求导数(polyder)

■ MATLAB 实现

键入　e=polyder(c)

得　e =   30   96   180   192   102

● 多项式求根(roots 和 poly 函数)

■ MATLAB 实现

键入　ra=roots(a); rb=roots(b); rc=roots(c); ra, rb, rc

得

```
ra = -1.6506
     -0.1747 + 1.5469i
     -0.1747 - 1.5469i
rb = -1.0000 + 1.4142i
     -1.0000 - 1.4142i
```

rc 是 ra 和 rb 的并，这是完全可以预计到的。

由根求多项式系数是 roots 的逆运算，其函数名也是 poly。

```
a = poly(ra);  b = poly(rb)
```

从 poly 函数的用法可以看出 MATLAB 的智能特点，当 a 是向量时，poly 把它看做根来组成多项式；当 a 是方阵时，poly 用它组成方阵的特征多项式。

● 多项式求值(polyval)　　将多项式 a 中的自变量 x 赋予值 xv 时，该多项式的值可用 F = polyval (a,xv)求得

其中 xv 可以是复数，也可以是矩阵或数组，此时 polyval 对输入变元作元素群运算，这对于求线性系统的频率特性特别方便。polyvalm 则对输入的变元阵（必须是方阵）作矩阵多项式运算。

【例 4.5】 设 **a** 为系统分母系数向量，**b** 为系统分子系数向量，求此系统的频率响应并画出频率特性。

解：■ MATLAB 实现

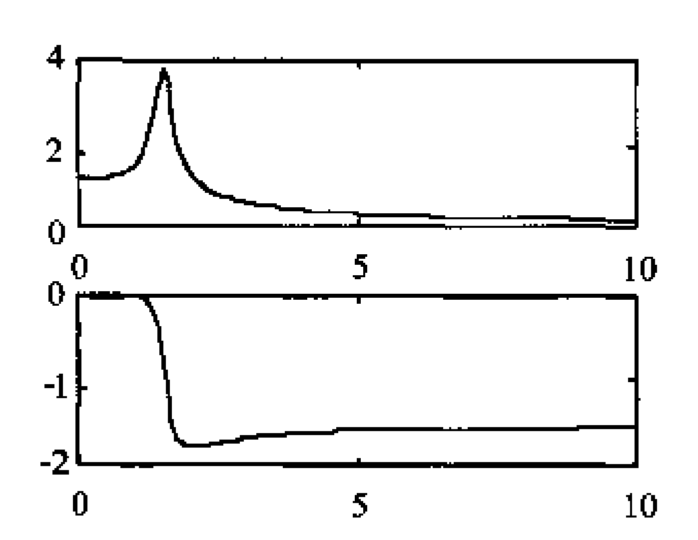
先令频率数组 w 取线性间隔：

```
w=linspace(0,10);              % 在 w=0 到 10 之间按线性间隔取 100 点（默认值）
A=polyval(a,j*w);B=polyval(b,j*w);     % 分别求分母分子多项式的值（为复数数组）
subplot(2,1,1);plot(w,abs(B./A)),      % 画两者元素群相除所得的幅频特性
subplot(2,1,2);plot(w,angle(B./A))     % 画相频特性
```

频率特性通常在对数坐标中绘制。因此，输入的频率数组取对数等间隔:

```
w1=logspace(-1,1)              % 在 w1 从 10⁻¹到 10 之间，按对数分割为 50 点（默认值）
F=polyval(b,j*w1)./polyval(a,j*w1);    % 求出这些点上的频率响应（复数）
subplot(2,1,1),loglog(w1,abs(F))       % 在双对数坐标中画出幅频特性
subplot(2,1,2);semilogx(w1,angle(F))   % 在双对数坐标(x)中画出相频特性
```

所得曲线见图 4.3。



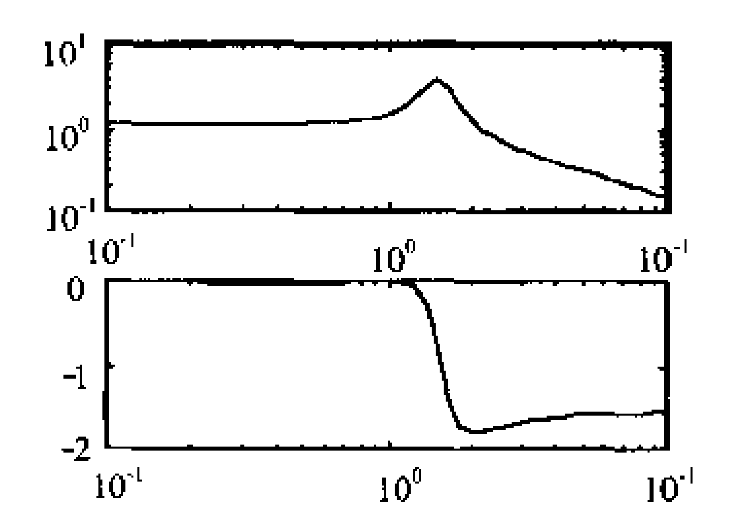


图 4.3 线性坐标和对数坐标中的频率特性

4.3.3 多项式拟合

➢ p=polyfit(x,y,n) 用于多项式曲线拟合。其中 x，y 是已知的 N 个数据点坐标向量，当然其长度均为 N。n 是用来拟合的多项式次数，p 是求出的多项式的系数，n 次多项式应该有 n+1 个系数，故 p 的长度为 n+1。拟合的准则是最小二乘法。

【例 4.6】 设原始数据为 *x* 时在 11 个点上测得的 *y* 值：

```
x=0:0.1:1;y=[-0.447,1.978,3.28,6.16,7.08,7.34,7.66,9.56,9.48,9.30,11.2];
```

■ MATLAB 实现

· 线性拟合：`a1=polyfit(x,y,1);`

求出 a1 后，可求出 xi=linspace(0,1);（即 100 个点）上的 yi1 值并绘图：

```
yi1=polyval(a1,xi); plot(x,y,'o',xi,yi1,'b'),pause
```

其中原始数据用圆圈标出，而拟合曲线为蓝色。依此类推，有

· 二次拟合：`a2=polyfit(x,y,2);yi2=polyval(a2,xi);plot(x,y,'o',xi,yi2,'m')`

· 三次拟合：`a3=polyfit(x,y,3);yi3=polyval(a3,xi);plot(x,y,'o',xi,yi3,'r')`

· 九次拟合：`a9=polyfit(x,y,9);yi9=polyval(a9,xi);plot(x,y,'o',xi,yi9,'c')`

· 十次拟合：`a10=polyfit(x,y,10);yi10=polyval(a10,xi);plot(x,y,'o',xi,yi10,'g')`

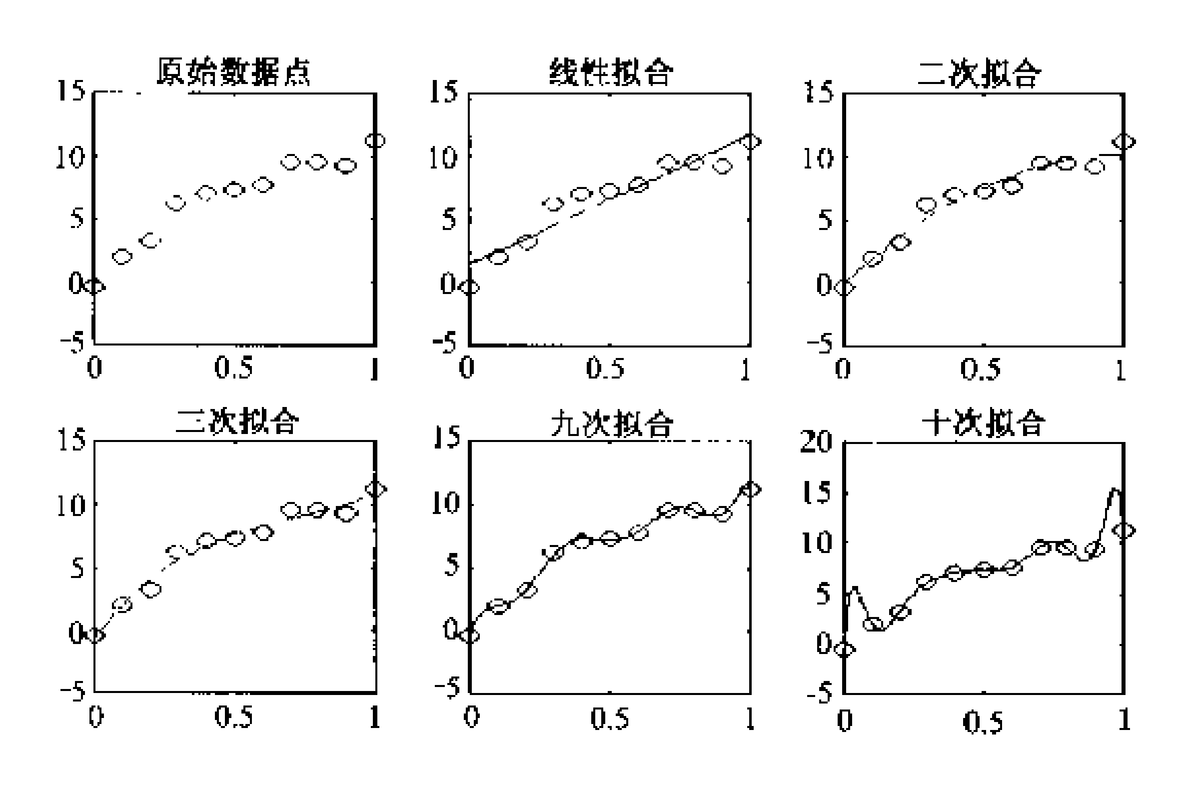


图 4.4　不同的逼近次数产生的不同曲线

所得的曲线见图 4.4。给定 11 点的最大拟合阶次为 10，此时拟合曲线将通过全部给定点。可以看出，拟合曲线的阶次太高会造成曲线振荡，反而看不出函数关系的基本规律，并不一定好。

4.3.4　多项式插值

插值和拟合的不同处有以下几点。

● 插值函数通常是分段的，因而人们关心的不是函数的表达式，而是插值得的数据点。插值数据应通过给定的数据点 x，y。插值函数一般地可表为：

➢　yi=interp1(x,y,xi，'method')　　其中 xi 为插值范围内的任意点集的 x 坐标，yi 是插值后的对应数据点集的 y 坐标。method 为插值函数的类型选项，有 linear（线性，默认项）、cubic（三次）和 cubic spline（三次样条）等三种。

1．一维插值函数 interp1

【例 4.7】 仍取例 4.6 中的 x，y，xi，求其线性和三次插值曲线。

■ MATLAB 实现

• 线性插值：yi1=interp1(x,y,xi);plot(x,y,'o',xi,yi1)

• 三次插值：yi2=interp1(x,y,xi,'spline');plot(x,y,'o',xi,yi2,'g')

所得曲线见图 4.5 及图 4.6，三次插值的结果比较光滑。

2．二维插值函数 zi=interp2(x,y,z,xi,yi,'method')

【例 4.8】 已知某矩形温箱中 3×5 个测试点上的温度，求全箱的温度分布。

给定：width=1:5;depth=1:3;temps=[82 81 80 82 84;79 63 61 65 81;84 84 82 85 86];

要求计算沿宽度和深度细分网格：di=1:0.2:3;wi=1:0.2:5；交点上的温度。

解：■ MATLAB 实现

```
tc=interp2(width,depth,temps,wi,di','cubic'); %求各点温度
mesh(wi,di,tc)  %画三维曲面
```

所得温度分布图形如图 4.7 所示。

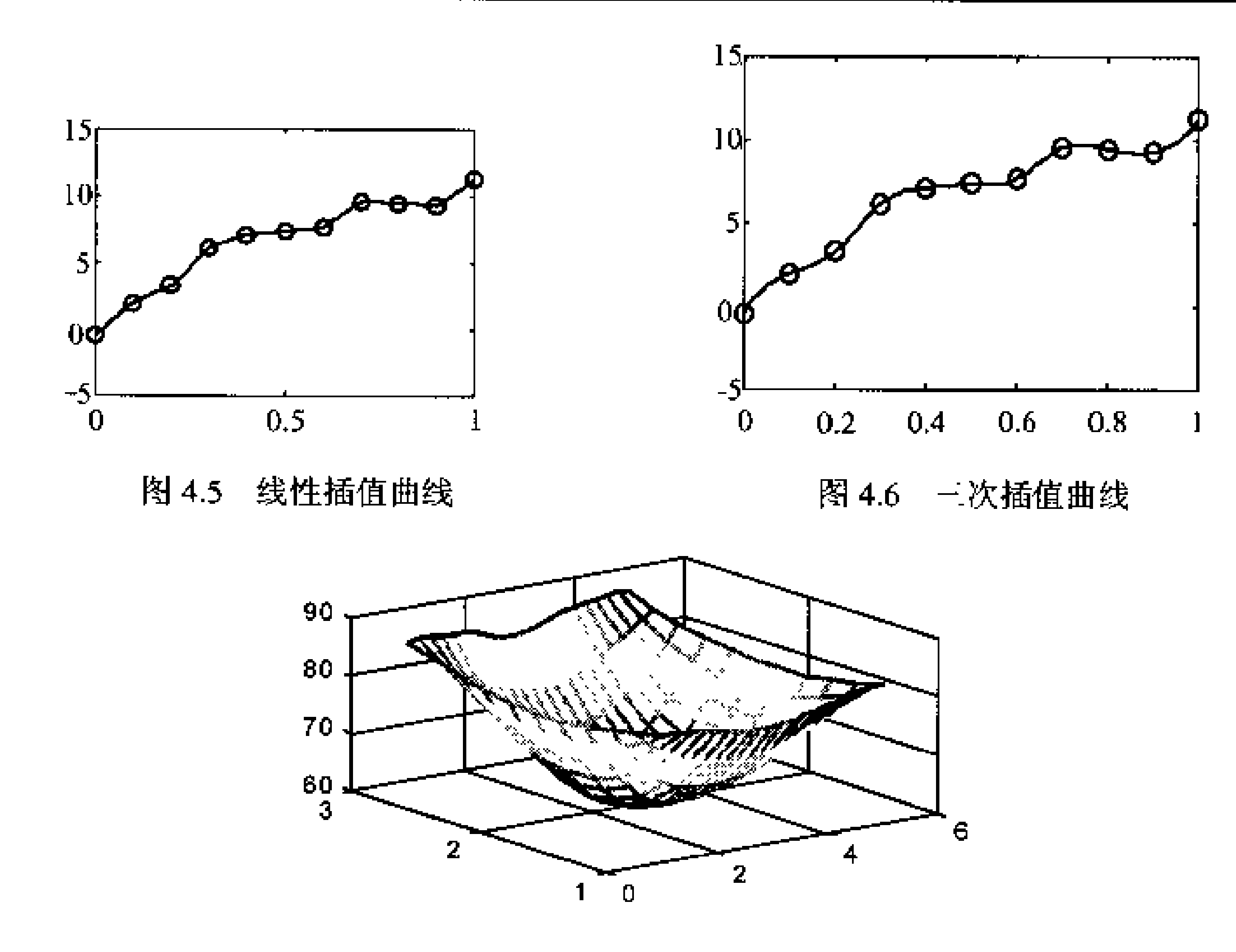


图 4.5　线性插值曲线

图 4.6　二次插值曲线

图 4.7　二维插值的曲面

注意　interp2 中所用的 wi，di'是宽度和深度方向的细分坐标向量，di 必须变换为列向量，插值函数运算时会自动将它们转变为宽度乘深度平面上的网格，并计算网格点上的温度。

4.3.5　线性微分方程的解(residue)

线性常微分方程的解可用拉普拉斯算子 s 表示为

$$Y(s)= B(s)/A(s)$$

其中 $B(s)$和 $A(s)$都是 s 的多项式，分母多项式的次数 n 通常高于分子多项式的次数 m。在时间域的解 $y(t)$是 $Y(s)$的拉普拉斯反变换。求反变换的重要方法之一是部分分式法，即将上述多项式分解为多个 s 的一次分式之和。留数函数 residue 可以完成这一任务。

步骤：

（1）用[r,p,k]= residue(b,a) 求出 Y(s)的极点数组 p 和留数数组 r（设分母比分子阶数高，故 k=0），因而 Y(s)可表为

$$Y(s)=\frac{r(1)}{s-p(1)}+\frac{r(2)}{s-p(2)}+\frac{r(3)}{s-p(3)}+\frac{r(4)}{s-p(4)}+\cdots$$

（2）求它的反变换。得

y(t) = r(1)*exp(p(1)*t) +r(2)*exp(p(2)*t) +r(3)*exp(p(3)*t) + r(4)*exp(p(4)*t) + …

【例 4.9】 求解线性常微分方程

$$y''' + 5y'' + 4y' + 7y = 3u'' + 0.5u' + 4u$$

在输入 $u(t)$为单位脉冲及单位阶跃信号时的解析解。

解：用 Laplace 变换（脉冲输入 $u(s)$=1，阶跃输入 $u(s)$=1/s）

$$y(s)=\frac{3s^2+0.5s+4s}{s^3+5s^2+4s+7}u(s)=\frac{b(s)}{a(s)}$$

■ 在脉冲输入时的响应

```
a=[1,5,4,7]; b=[3,0.5,4]; [r,p,k]=residue(b,a)
```

得

```
r =   3.2288
     -0.1144 + 0.0730i
     -0.1144 - 0.0730i
p =  -4.4548
     -0.2726 + 1.2235i
     -0.2726 - 1.2235i
k =[ ]
```

求时域解，先设定时间数组 t=0:0.2:10；然后列出：

```
yi=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t)+r(3)*exp(p(3)*t); plot(t,yi)
```

■ 在阶跃输入时的响应（此时分母由于乘了个 s，a 将提高一阶，右端多加一个零）；

```
a=[1,5,4,7,0]; b=[3,0.5,4]; [r,p,k]=residue(b,a)
r =  -0.7248
      0.0767 + 0.0764i
      0.0767 - 0.0764i
      0.5714 + 0.0000i
p =  -4.4548
     -0.2726 + 1.2235i
     -0.2726 - 1.2235i
      0
k =      [ ]
ys=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t)+r(3)*exp(p(3)*t)+r(4); plot(t,ys)
```

所得曲线见图 4.8。

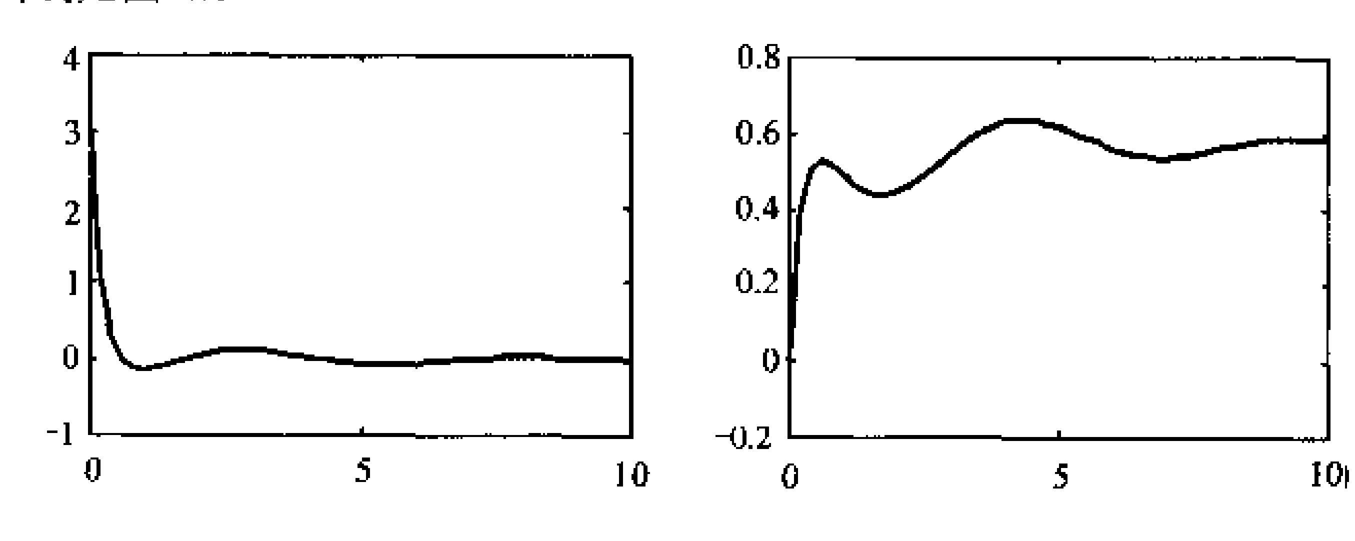


图 4.8　脉冲过渡响应和阶跃过渡响应

MATLAB 中的这一类命令有一个共同特点，其输入变元不仅有矩阵变量，而且有函数名。在执行这些命令时，要不断地调用函数作为输入变元。因此，完成这一类命令的功能比较丰富而灵活。掌握了它就能编写出更为高明的程序。

4.4　函数功能和数值积分函数库(funfun)

4.4.1　函数功能和数值积分函数库的主要子程序

函数功能和数值积分函数库的主要子程序见表 4.5，将它分为两类来探讨。第一类是对任意非线性函数的分析，包括求极值，过零点等；第二类求任意函数的数值积分，包括定积分和微分方程的数值解等。它们的共同特点是必须会自行定义函数。

表 4.5　函数功能和数值积分函数库（funfun）(e)

	函 数 名	功 能 说 明	需调用的函数
最优化和求根	fmin	单变量函数求极小值 y_{min}	给出待分析的函数 $y=f(x)$
	fmins	多变量函数求极小值	
	fzero	单变量函数求 $y=0$ 处的 x	
数值定积分	quad	数值积分计算（低阶）	给出被积分的函数 $f(x)$dy/dx=$f(x)$，求 y
	quad8	数值积分计算（高阶）	
	dblquad*	双精度数值积分	
函数绘图	ezplot*	简便的函数绘图器	
	fplot	画函数曲线 $y=f(x)$	
内 联 (INLINE) 函数对象	inline	构成 INLINE 函数对象	
	argnames	变元名	
	formula	函数公式	
	char	把 INLINE 函数转换为字符数组	
	vectorize	使字符串或 INLINE 函数向量化	
常微分方程数值积分器	ode45	解非刚性微分方程（中阶方法）	给出导数的函数表达式 dy/dx=$f(y,x)$，求 y
	ode23	解非刚性微分方程（低阶方法）	
	ode113*	解非刚性微分方程（变阶方法）	
	ode15s*	解刚性微分方程（变阶方法）	
	ode23s*	解刚性微分方程（低阶方法）	
	odefile	ODE 文件语法	
ODE 输出函数	odeplot*	时间序列	
	odephas2*	ODE 输出函数的二维相平面	
	odephas3*	ODE 输出函数的三维相空间	
	odeprint*	打印 ODE 输出函数	

下面取本书 2.6 节中定义过的函数 humps.m 来说明这些子程序的用法。它定义了下列非线性函数：

$$y=\frac{1}{(x-0.3)^2+0.01}+\frac{1}{(x+0.9)^2+0.04}-6$$

4.4.2　非线性函数的分析

■ 绘制函数曲线 fplot

其格式为：fplot('函数名'，[初值 x0，终值 xf])，

例如，要画出 humps 函数在 x=0～2 之间的曲线，可键入

```
fplot('humps',[0,2]),grid
```

得出图 4.9 所示的曲线。

fplot 函数对于快速了解一些复杂特殊函数的波形很有用处。例如，求其中第一类 Bessel 函数（见表 4.5），可用

```
fplot('besselj(alpha,x) ',[0,10])
```

设 alpha 为 1，2，5 时，得到图 4.10 所示的曲线。

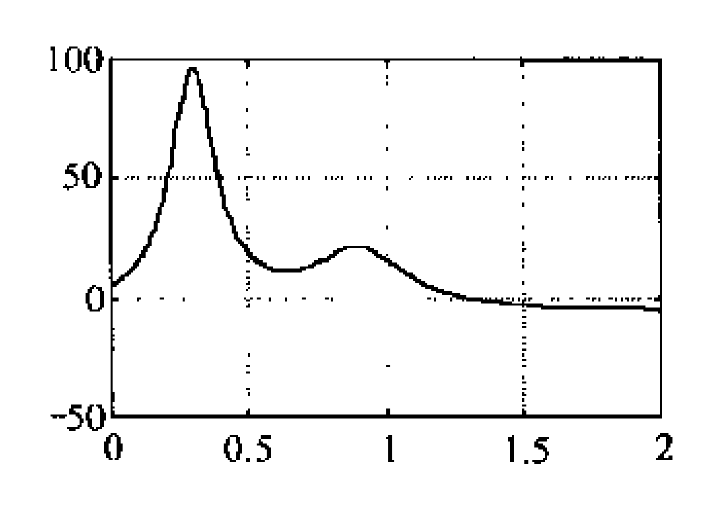


图 4.9　humps 函数的曲线

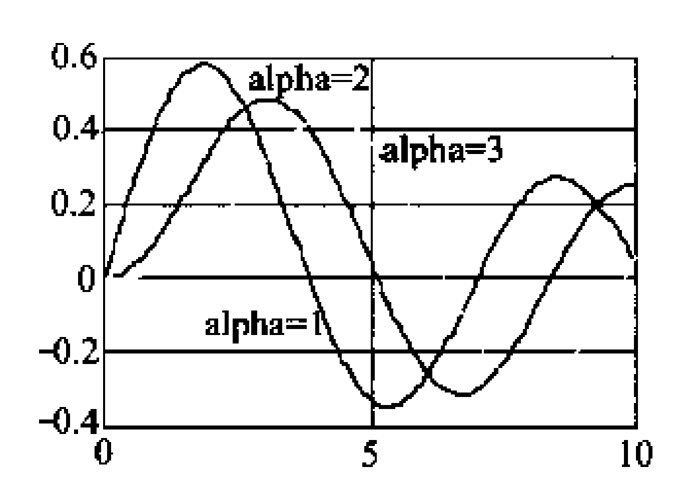


图 4.10　第一类 Bessel 函数的曲线

■ 求函数极值 fmin

其格式为：fmin('函数名'，初值 x0，终值 xf)，

例如，求 humps 函数在 x=0～1.5 之间的极小值，则

键入　　m=fmin('humps',0,1.5)

得　　m =　　0.6370

■ 求函数零点 fzero

其格式为：fzero('函数名'，初猜值 x0)

例如，求 humps 函数在 x=1 附近的过零点，则

键入　　z=fzero('humps',1),

得　　z =　　1.2995

以上给出的是这些函数调用的典型格式，还有其他选项可作为变元，例如

```
fplot('tan',[-2*pi 2*pi -2*pi 2*pi],'*'),grid
```

在第 2 项变元中增加了 y 轴的上下限，第 3 项变元是线型。所得图形见图 4.11(a)，读者可从 help fplot 中得到进一步的信息。

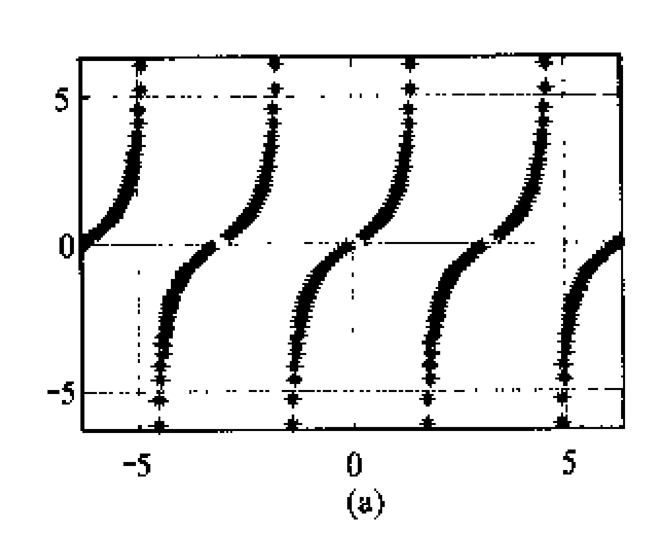
MATLAB 5.x 中还新增了一个简便画函数图的命令 ezplot(读作 easy plot)，它连自变量范围都无需规定，其默认的自变量范围为[-2π,2π]。因此只要键入

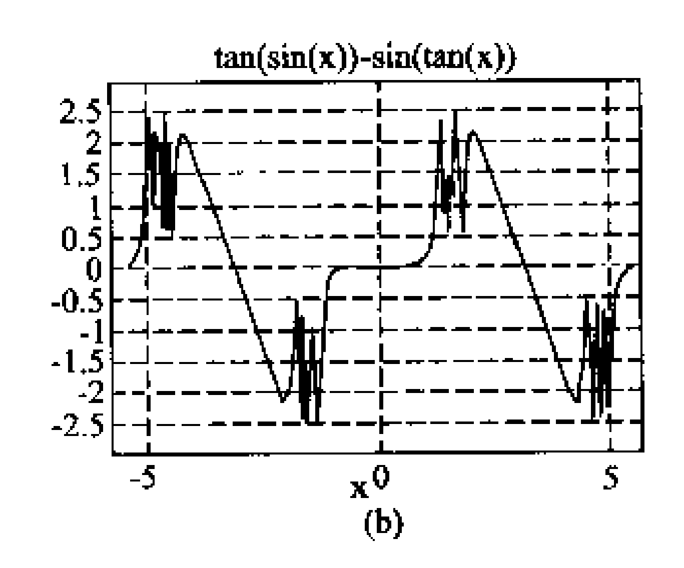
```
ezplot tan(x),grid
```

也可得到类似于图 4.11(a)的曲线，只是*号变为了实线。若键入

```
ezplot tan(sin(x))-sin(tan(x))
```

所得图形见图 4.11(b)。可以看出，图上还自动作出了标注。


(a)



(b)

图 4.11　由 fplot 和 ezplot 画出的曲线

4.4.3　任意函数的数值积分

- 定积分子程序(quad 及 quad8) 的格式为：

➢ quad（'函数名'，初值 x0，终值 xf），

例如，求 humps 函数在 x=1～2 之间的定积分。

键入　　s=quad('humps',1,2)

得　　　s =　-0.5321

不难用定积分函数来求不定积分的数值解。只要固定积分下限，用 for 循环，把积分上限逐步增加即可。

例如要求 humps 函数以 x=0 为下限的不定积分，可编写下列程序

```
for i=1:20
    x(i)=0.1*i;
y(i)=quad('humps',0,x(i));
end,  plot( x,y)
```

得出的曲线如图 4.12 所示。可以与图 4.9 对照，确认它是 humps 曲线的积分。

图 4.12　humps 函数的积分曲线

- 微分方程数字解(ode23，ode45 等)

如果微分方程可化为一阶微分方程组的形式：

$$dy/dx = f(x,y)$$

其中，x 是标量，y 可以是一个列向量。

$f(x,y)$是以 x，y 为变元的函数，用 MATLAB 函数文件表述，设文件名为 yprime.m，则求此微分方程的数值解的子程序调用格式为：

➢ [x,y] = ode23('yprime'，自变量初值 x0，自变量终值 xf，因变量初值 y0)

对于 humps 函数，不能直接用 ode23 做数值积分，其原因在于 humps 只有一个输入变元 x。而微分方程数值解的函数 ode23，要求被调用的函数有两个输入变元。如果把 humps 函数文件加一个虚的变元 y，即把它的第一句换成 function yp = humps1(x, y)，并将此函数另存成一个 humps1.m 文件，则

[x,y] = ode23('humps1', 0, 2, 1)；plot(x,y)

表示在初值 y0=1 的条件下，从 x0=1 到 xf=2 求微分方程的数值解，运行后可以得到与图 4.12 相仿的曲线，只是向上平移了一个单位，因为这里设了 y0=1。在这个例子中，函数

humps1 中的 y 只是一个虚的变元，比较简单。

【例 4.10】 求下列微分方程（范德堡方程）的数值解。

$y'' + r(y^2-1)y' + y = 0$

解：■ 分析

它可写成导数在左端的两个一阶微分方程构成的方程组。

$$y_1' = y_2$$

$$y_2' = r(1-y_1^2)y_2 - y_1$$

先要建立反映此微分方程组右端的函数文件 vdpl.m，存入子目录 user 中，其内容为

```
function  yprime = vdpl (x,y)
global  r            % r 值由主程序通过全局变量传送
yprime = [ y(2);  r*(1 - y(1).^2).*y(2) - y(1) ];    % 两行单列向量
```

■ MATLAB 主程序

```
global r , r =input ('输入 r, 在 0<r<10 之间选择')
x0=input('x0=' ); xf= input( 'xf=' ); y0=input( 'y0=[y10;y20]= ' );
[x,y] = ode45( 'vdpl' , x0, xf, y0); plot(x, y)
```

在 r=2, x0=0, xf=30, y0=[1;2]条件下得出的曲线如图 4.13 所示。

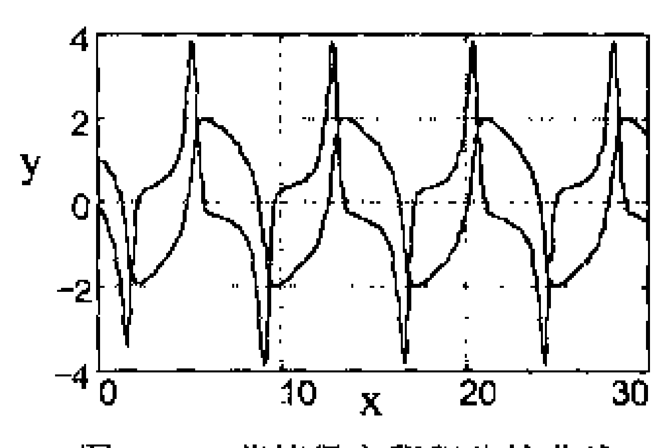


图 4.13　范德堡方程积分的曲线

ode45 是高阶的数值积分函数，其步长可以取得较大，并能保证较高的精度。其调用方法与 ode23 相仿。这些函数都有自动选择步长的功能，以保证把误差控制在 0.001 以下。如果要改变容许误差 tol，可在输入变元中增加其他选项，详情可从 help 命令中获得。在 MATLAB5.x 中，求微分方程数值解的函数有了进一步的扩展，增加了 ode23s，ode15s，ode113 等数值积分函数，并增加了输出绘制相平面曲线的命令 odephas2 和 odephase3 等。

从本节可见，要利用函数功能，就要会定义各种复杂函数，表 4.6 列出了 MATLAB 中已定义的某些复杂的特殊函数。

表 4.6　特殊函数库(specfun) (u)

特殊数学函数	airy *	Airy 函数	bessely	第二类 Bessel 函数
	besselj	第一类 Bessel 函数	besselh *	第三类 Bessel 函数(Hankel 函数)
	besseli	第一类修正的 Bessel 函数	besselk	第二类修正的 Bessel 函数
	beta	Beta 函数	betainc	不完全的 Beta 函数
	betaln	Beta 函数的对数	ellipj	Jacobi 椭圆函数
	ellipke	完全椭圆积分	erf	误差函数
	erfc	误差补函数	erfcx	标定的误差补函数
	erfinv	逆误差函数	expint	指数整数函数
	gamma	伽马函数	gammainc	不完全的伽马函数
	gammaln	伽马函数的对数	legendre	联合的 Legendre 函数
	cross	向量叉乘		

续表

数论函数	factor	素数分解	primes *	产生素数清单
	gcd	最大公约数	lcm	最小公倍数
	rat	有理分式近似	rats	有理分式输出
	isprime *	是素数时为真	perms	所有可能的排列数
	nchoosek *	N 取 K 的组合数		
坐标变换	cart2sph	从笛卡尔向球坐标变换	cart2pol	从笛卡尔向极坐标变换
	pol2cart	从极坐标向笛卡尔坐标变换	sph2cart	从球坐标向笛卡尔坐标变换

4.5　字符串函数库(strfun)

MATLAB 的程序和标识符都是用字符串来表示的。每个字符有它对应的 ASCII 码。4.4 节中的函数，都把字符串当做变元来看待，从而使程序更为简化和高效。有时也需要把字符串当做数码来处理。字符串函数库（见表 4.7）中的命令，都是为了增强这一功能。对初学者来说，这部分并不重要，但若要编写出人机界面优良，能调用各种函数和文件的高级程序，字符串函数库则必不可少。

表 4.7　字符串函数库（strfun）(v)

一般函数	char	建立字符数组（字符串）	blanks	空格字符串
	double	把字符串转换为数字	deblank	去除尾部的空格
	cellstr	由字符数组组成字符阵列	eval	执行程序字符串
测试	ischar	是字符串时为真	isletter	是英文字符时为真
	iscellstr	是字符阵列时为真	isspace	是空格字符时为真.
字符串比较	strcmp	字符串比较	strncmp	比较前 N 个字符串
	findstr	在字符串中找另一字符串	strjust	调整字符串
	upper	将字符串变为大写	strmatch	找到可能的匹配字符串
	lower	将字符串变为小写	strrep	用一个字符串替代另一个
	strcat	链接字符串	strtok	在字符串中找一个令牌
	strvcat	竖向链接字符串		
字符串与数的转换	num2str	把数转换为字符串	str2mat	由单个字符串形成文本矩阵
	int2str	把整数转换为字符串	sprintf	在格式控制下把数转换为字符串
	str2num	把字符串转换为数	sscanf	在格式控制下把字符串转换为数
	mat2str	把矩阵转换为字符串		
基数的转换	hex2num	把十六进制字符串转换为 IEEE 浮点数	dec2bin	把十进制整数变换为二进制字符串
	hex2dec	把十六进制字符串转换为十进制整数	base2dec	把基数 B 字符串变换为十进制整数
	dec2hex	把十进制整数转换为十六进制字符串	dec2base	把十进制整数变换为基数 B 字符串
	bin2dec	把二进制字符串变换为十进制整数		

4.5.1　字符串的赋值

语句 s=′abxyABXY0189′ 把字符串赋值给 s，其结果是

s = abyzABYZ0189

而　　size(s)= 1　12

说明它是以行向量的形式存储的。当然它内部带有字符串的标志，故在屏幕上显示出字符。要找到 s 所对应的 ASCII 码，可用 abs 命令：

abs(s)= 97　98　121　122　65　66　89　90　48　49　56　57

从中可以知道英文大小写字母和十进制数字的 ASCII 码值。用 setstr 命令可作逆向变换：

setstr(abs(s)) = abyzABYZ0189

要求出字母和数字的十六进制 ASCII 码值，可用 dec2hex 命令：

dec2hex(abs(s))= 61 62 79 7A 41 42 59 5A 30 31 38 39

MATLAB 显示时并没有各码之间的空格，这里加上空格是为了便于读者阅读。

可以把几个字符串沿行向串接，构成更长的字符串，如

键入　s1=['welcome ', s]

得　s1 = welcome abyzABYZ0189

可以把几个长度相同的字符串沿列向并列，组成一个字符串矩阵，如

s2 =['a=5　　';'b=2　　';'c=a+b*b']

这时必须在前两个字符串中增添若干个空格，保证三个字符串长度均为 7，否则赋值无效。

4.5.2 字符串语句的执行

如果字符串的内容是 MATLAB 语句，如上述的 s2，则可以用 eval 命令来使它执行。如

键入　for k=1:3 eval(s2(k,:)), end

结果为　a=5, b=2, c=9.

在编写 MATLAB 的演示程序时，往往要通过人机交互，让用户输入某种表达式（而不只是数据），然后按此表达式执行，这种程序的格式如下：

```
st = input(' s=表达式 ', ' s '); eval(st)
```

input 语句中的' s ' 表示把输入当做字符串来接受，因此，用户键入的字符串就不必加引号了。下面的例子说明如何将 eval 函数和 load 函数一起使用，读出 10 个具有连续文件名 mydata1, mydata1，…, mydata10 的数据文件：

```
for i=1:10 fname='mydata'; eval(['load ',fname,int2str(i)]), end
```

特别注意 int2str(i)把数 1:10 转换为字符 1:10。在显示屏上看不出两者有什么差别，但要记住字符 0:10 的 ASCII 码是 48:57。数 10 只占一个 MATLAB 双精度存储单元，而字符串 10 却占两个存储单元，并构成一个单行两列的矩阵。在 eval 后的输入变元必须是一个构成 MATLAB 语句的字符串。因此，必须把三个字符串接起来，并且不要忘掉在 load 后面加一个空格。eval 命令是在较高级的程序中常常遇到的，要学会使用。

4.5.3 字符串输入输出

前面一直用 disp 函数来进行字符串和数据的输出。disp(' pi=')将显示引号内的字符串，此处为 pi=。disp(pi)将显示变量 pi 的值 3.1416 或其他的 8 种显示格式之一，由 format 命令确定。想把字符串 pi=和变量 pi 的值显示在一行上，试用 disp(' p i=')，结果显示这是非法的。这时应该用 sprintf 函数，它可把数据按要求的格式转换为字符串，再把它与需要显示

的字符串组装成一个长字符串，使显示格式非常灵活，人机界面更为友好。

键入　st=sprintf('圆周率 pi= %8.5f',pi);disp(st)

结果为：　圆周率 pi=　3.14159

其中%为数据格式符，f 表示十进制浮点，8.5 表示数字的长度为 8 位，小数点后 5 位。从%到 f 之间的字符都是不显示的，它只规定了显示数据 pi 的格式。

【例 4.11】 再举一个用 sprintf 的例子。

```
x = 0:10:90; y = [x; sin(x*pi/180)];disp(sprintf('%10.2f  %12.8f\n',y))
    0.00      0.00000000
   10.00      0.17364818
   ...        ...
   80.00      0.98480775
   90.00      1.00000000
```

sprintf 命令是从 C 语言中的同名命令演化来的 。sscanf 则是它的逆命令。相仿的还有 fprintf 和 fscanf（见表 3.3）。

4.6　稀疏矩阵函数库(sparfun)

在许多科学和工程计算中，人们会遇到很大的矩阵，例如几千行几千列，元素则数以百万计。而这些元素中，绝大多数为零。这反映了复杂事物中的诸变量之间，有直接关联的是少数。为了节省内存和提高计算速度，产生了稀疏矩阵的理论和方法。MATLAB 的稀疏矩阵函数库（见表 4.8）就是为这个目的开发的。稀疏矩阵只存储矩阵的非零元素，其表示形式如下例

```
设      x =   0          0         0        0         0
              0          0         0        0.1302    0
              -0.5936    0         0        0         0
              0          1.4499    0.0388   0         0
              0.5589     0         0        0         -0.0954
```

- sparse 命令把完全矩阵转换为稀疏矩阵，

```
键入    s=sparse(x)
得      s =(3,1)      -0.5936
           (5,1)       0.5589
           (4,2)       1.4499
           (4,3)       0.0388
           (2,4)       0.1302
           (5,5)      -0.0954
```

括号内是非零元素的行号和列号，后面则是元素的值。可见 25 个元素中它只需保存 6 个。用命令 whos 检验 MATLAB 工作空间中的变量存储情况，结果如下：

Name	Size	Elements	Bytes	Density	Complex
s	5 by 5	6	92	0.2400	No
x	5 by 5	25	200	Full	No

可见 s 所占存储单元数为 x 所占存储单元数的 0.24 倍。

表 4.8 稀疏矩阵函数库(s)

初等稀疏矩阵	speye	稀疏单位矩阵	sprandn	正态分布稀疏随机矩阵
	sprandsym	稀疏对称随机矩阵	spdiags	由对角矩阵生成稀疏矩阵
	sprand	均匀分布稀疏随机矩阵		
全矩阵向稀疏矩阵的转换.	sparse	从非零元素创建稀疏矩阵	full	变稀疏矩阵为全矩阵
	find	查找非零元素的下标	spconvert	稀疏矩阵的外部格式转换
用稀疏矩阵的非零元素工作	nnz	非零元素的数目	nonzeros	非零元素
	nzmax	分配给非零元素的内存总额	spones	用 1 替换非零元素
	spalloc	为非零元素分配内存	issparse	矩阵为稀疏时得真
	spfun	对非零元素施加函数	spy	显示稀疏结构
重组算法	colmmd	列的最小度	symmmd	最小对称度
	symrcm	CuthillMcKee 逆排序	colperm	按非零元素数目对列排序
	randperm	随机交换向量	dmperm	DulmageMendelsohn 分解
线性代数	luinc *	不完全 lu 分解	svds *	一些奇异值
	sprank	结构的秩		
线性方程(迭代方法)	pcg *	预设条件的共轭梯度法	bicg *	双共轭梯度法
	bicgstab *	双共轭梯度稳定法	cgs *	共轭梯度平方法
	gmres *	通用最小残差法	qmr *	准最小残差法
对树的运算	treelayout	对单树或多树的布局	Treeplot	绘出结构树
	etree	矩阵的消除树	etreeplot	绘出消除树
	gplot	按图论方法画图		
杂项	symbfact	Symbolic 符号因式分解	spparms	为稀疏矩阵设定参数
	spaugment	形成最小二乘增广系统		

- full 命令把稀疏矩阵转换为完全矩阵。初等矩阵运算的命令，如四则运算，求逆等均可直接用于稀疏矩阵，矩阵分解等命令则要把它化为完全矩阵后才能调用。在大学本科中，解电路矩阵方程也会遇到大量系数为零的情况。但课程习题所遇到的阶次不会高，完全矩阵足以应付。用到稀疏函数库的可能性不大，本书只把函数库列出备查。读者遇到这类应用时，得先参阅有关稀疏矩阵理论的书籍，再用 MATLAB 中的 help 文本或参看其他参考书。

4.7 图形界面函数库(Guitools)

MATLAB 5.x 中的图形界面函数库是为了设计像 demo 程序中那样的界面的。通常，在设计了一个较丰富的 MATLAB 函数集之后，为了便于他人使用，应该避免让用户去记忆和键入这些函数的名称。将其做成这样的图形界面，其中有计算机向用户显示结果或出错信息的各种对话框；有用户对计算机实行控制的各种按钮等等。MATLAB 有一本说明书，专门介绍图形界面的设计方法，其搜索路径为：

```
MATLAB\help\pdf_doc\matlab\Buildgui
```

它所用的函数，都已包括在表 4.9 中。读者将来工作中若有需要，可以找此说明书参阅。

表 4.9 图形用户界面工具函数库(Guitools)* (x)

GUI函数	uicontrol	建立用户控制菜单	dragrect	用鼠标拖引方框
	uimenu	建立用户界面菜单	rbbox	橡皮擦方框
	ginput	用鼠标输入图形	waitfor	执行模块并等待事件发生
	selectmoveresize	交互地选择、移动、变形或拷贝对象	waitforbuttonpress	等待键或按钮在图上按下
	uiwait	执行模块并等待继续命令	uiresume	继续执行模块 M 文件
GUI设计工具	guide	设计 GUI	Menuedit	编辑菜单
	align	对齐 UI 控制和轴线	propedit	编辑特性
	cbedit	编辑 Callback		
对话框	dialog	建立对话图形	warndlg	警告对话框
	axlimdlg	对话框轴线范围	uigetfile	标准的打开文件对话框
	errordlg	错误对话框	uiputfile	标准的存储文件对话框
	helpdlg	帮助对话框	uisetcolor	对话框颜色选择
	inputdlg	输入对话框	uisetfont	对话框字体选择
	listdlg	列出待选对话框	pagedlg	对话框页面位置选择
	menu	生成用户输入选择菜单	printdlg	打印对话框
	msgbox	消息框	waitbar	显示等待条
	questdlg	问题对话框		
菜单设施	makemenu	建立菜单结构	umtoggle	改变用户界面菜单对象的检查状态
	menubar	为不同的计算机设定菜单条默认值	winmenu	为 Window 菜单项建立子菜单
工具条按钮群函数	btngroup	建立工具条按钮群	btnstate	工具条按钮群的排队状态
	btnpress	为工具条按钮群按动管理器	btndown	在工具条按钮群中按下按钮
	btnup	在工具条按钮群中抬起按钮		
用户定义	clruprop	清除用户定义特性	setuprop	设定用户定义特性
	getuprop	获取用户定义特性		
杂项函数	allchild	获取所有子对象	popupstr	获取弹出菜单选择字符串
	findall	寻找所有对象	remapfig	变换图形对象的位置
	hidegui	隐藏/不隐藏 GUI	setptr	设定图形指针
	edtext	对轴系文本对象作交互的编辑	setstatus	设定图形中的状态文本字符串
	getstatus	获取图形中的状态文本字符串	overobj	获取指针过去后的对象句柄
	getptr	获取图形指针		

4.8 数据类型函数库(datatypes)

由于计算机应用十分广泛，要它处理的数据类型很多，仅以数为例就有整数型(0~65535，16 位)、带符号整数型(−32768~32767，15 位加符号位)、字符型（0~255，8 位)、浮点单精度型（32 位)、浮点双精度型（64 位）等等，其他还有字符类型、指针类型等。在其他语言(例如 C 语言)中，编程时对每一个变量都要

作出规定，这就增加了不少语句，而且要好好思考计划，不然容易出错。

在 MATLAB 中，所有的数都用一种浮点双精度类型来存储和运算。因而省略了定义类型的语句，编程时无需去思考分辨，也减少了错误。当然对于那些本来就只要用一两个字节来表示的变量来说，这种做法既浪费内存，又降低了运算速度。但用牺牲（存储）空间和（运算）时间来换取人机交互友善性的战略被证明是有效的，它形成了科学计算语言的特色，使人们不在编程的细节上化精力，而把注意力集中到科学计算的方法和建模合理性等大问题上去。

在工程和管理系统中，常常需要分层次地把一些不同类型、不同尺寸和不同分层的数据组织起来，成为一个变量。MATLAB 5.x 专门为此定义了两种数据类型，一种称为结构阵列，另一种称为单元阵列。这两种数据类型在其他语言中很少见，颇具特色，在此做一简单介绍。

4.8.1　结构阵列

假如要为一个班的学生建立一套管理档案 student，记录每个学生的三个项目：姓名（字符串）、出生日期（数字）、四门课（德育，数学，语文，体育）的成绩（数组）。这三项内容的数据类型各不相同，姓名的长度也不同，可以用在 student 后加域（field）的方法来建立一个结构阵列。

键入值
```
student.name='John';
student.birthday='1985.06.15';
student.score=[85,78,92,68];
```

再键入
```
student
```

得到
```
ans =    name: 'John'
birthday: '1985.06.15'
score: [85.00 78.00 92.00 68.00]
```

也可用struct命令来建立结构阵列。
```
student(2)=struct('name','Alice','birthday','1986.01.20','score',…
[77,81,65,91]);
```

再键入
```
student
```

得到
```
student =
1×2 struct array with fields:
    name
    birthday
    score
```

如果要得到student各个分量的详细内容，

键入
```
for i=1:2   disp(student(i)),   end
```

也可以提取各个域的内容，

键入
```
student.score
```

得到
```
ans =        85.00        78.00        92.00        68.00
ans =        77.00        81.00        65.00        91.00
```

结构阵列也可以嵌套。例如若有的分数用优、良、中等级表示，则也必须把课程成绩用结构阵列来表示成为 student.score.molarity、student.score.math、student.score.lang 等等。

4.8.2　单元阵列

单元阵列是用来分层次地组织不同类型数据成为一个变量的另一种方法，与结构阵列的不同在于它不用域名，而是像矩阵那样用下标，与数值或字符串矩阵的区别在于用花括号代替方括号。例如，上述 student 也可用单元阵列来存储。

键入
```
name={'John';'Alice'};
birthday={'1985.06.15';'1986.01.20'};
score={[85,78,92,68];[77,81,65,91]};
student={name;birthday;score};
```
再键入　student

得到
```
student = {2×1 cell}
          {2×1 cell}
          {2×1 cell}
```
键入　student{1}

得到
```
ans= 'John'
'Alice'
```
键入　student{1}{2}

得到　ans='Alice'

键入　student{3}{2}

得到　ans =　77　81　65　91

键入　student{3}{2}(2:3)

得到　ans =　81　65

最后的一个是圆括号，因为到这一级是按数组定义的，而前两级都是按单元阵列定义的。

同一个例子，可以用不同数据类型，也可以用不同的分级方式：

```
(1) student----------姓名----------人
                   ---生日
                   ---成绩
(2) student---------人-----------姓名
                              ---生日
                              ---成绩
```

其对应的分级构造见图 4.14。

前一种方法在输入新人的时候不大方便。如果要把它改成后一种，即使得：
```
student{1}{1}= 'John'
student{1}{2}= '1985.06.15
student{1}{3}=  85  78  92  68
```
那就应该按下述语句输入
```
student{1}={'John';'1985.06.15';[85,78,92,68]}
student{2}={'Alice';'1986.01.20';[77,81,65,91]}
```
再键入　student

得　student =　　{3 × 1 cell}　　{3 × 1 cell}

即变量 student 由两个 3×1 的单元阵列组成，该单元阵列的三列分别为姓名、生日和成绩。

而键入　student {1}

得　ans =　　'John'

'1985. 06. 15'

[1 × 4 double]

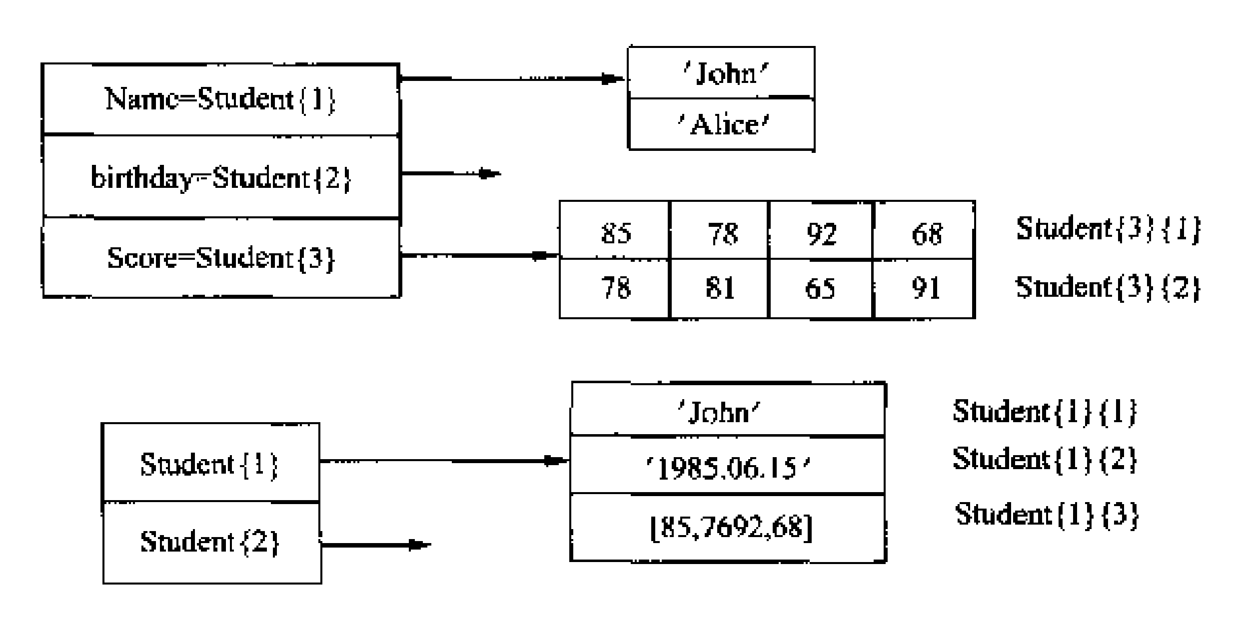


图 4.14　两种不同的数据构造方法

由此可见，结构阵列和单元阵列具有相似的功能。其差别在于使用的习惯。结构阵列利用域名来存储和调用各个元素，而单元阵列则是利用下标。前者的好处是具有明确的意义，便于记忆；而后者则比较简洁。

4.8.3　类和对象

所谓"类"就包含了对一种特定的变量结构及其运算方法的定义。所谓"对象"就指一种特定"类"中的某个变量或某个实例。所谓"面向对象的编程"就是描述如何利用"类"和对"对象"进行编程的方法。

MATLAB 的基本部分有 5 种固有（built-in）的类：

- Double　——浮点双精度类
- Sparse　——稀疏矩阵类
- Char　——字符串阵列类（char, 用双字节存储）
- Struct　——结构阵列类
- Cell　——单元阵列类

随着应用领域的扩大，某些 MATLAB 工具箱中也增加了一些其他的数据类型，例如为图像处理用的 unit8 类型(用单字节存储)，符号推理工具箱提供了 sym 类等等。结构阵列和单元阵列的应用大大扩展了变量的类型。比如上述的变量 student 就可以看做一种新的变量类型。

这些数据类的运算规则和原来为双精度类型设计的 MATLAB 命令和函数当然不同，所以它们原则上不能使用+、−、*、/、\ 等运算符和其他许多函数。MATLAB 采取了扩展运算符的方法，就是在采用一种特定的数据类型时，可为该种类型专门定义相应的运算符，例如用 plus 代替+，mtimes 代替*等（参见表 4.10 的最后部分），这些函数针对不同的数据类型，因此它们被存储在不同数据类型的子库中。

比如键入　help plus

在对 plus 函数作出说明以后，最后显示：

```
Overloaded methods
   help polynom/plus.m
   help sym/plus.m
   help zpk/plus.m
   help tf/plus.m
   help ss/plus.m
```

表示系统中对 polynom，sym，zpk，tf，ss 这 5 种数据类型分别由 plus.m 文件定义了 plus 运算符。其中 zpk，tf，ss 3 种数据类型用于控制系统工具箱中，后面将会介绍。polynom 和 sym 两种数据类型分别用于多项式和符号运算中。在 4.3 节中已经看到，多项式的加法不能简单用矩阵加法，而要单独编一个子程序 polyadd 来完成。多项式的乘法也不能简单用矩阵乘法，而要用 conv 来实现，如果把 polyadd 和 conv 放在 polynom 类的方法库中，并分别命名为 plus 和 mtimes，那么就可以用+号和*号来对多项式作运算了。MATLAB 在执行程序时，将先对被运算的变量进行检验，如果确认其属于 polynom 类别，就会调用方法库中的运算扩展符函数 polyadd 和 conv 来执行+号和*号运算符。特定的变量类型加上专门对这种变量类型进行运算的函数，就构成了一个新的类。有关类和对象的概念，在入门和基本应用中还不必使用，本书将在控制系统工具箱中用实例加以说明。

表 4.10 给出了数据类型和结构函数库。

表 4.10 数据类型和结构函数库(datatypes) * (b)

数据类型	double*	变换为双精度	sparse *	建立稀疏矩直接
	char *	建立字符（数组）串	cell *	建立单元阵列
	struct *	建立或变换为结构阵列	uint8 *	变换为无符号 8 位整数
	inline *	构成内联(INLINE)对象		
多维数组函数	cat *	链接数组	ndims *	维数
	ndgrid *	生成 N 维函数的阵列并插值	permute *	重新排列矩阵维数
	ipermute *	反向重排矩阵维数	shiftdim *	移动维数
	squeeze *	去除单项维数（降维）		
单元阵列函数	celldisp *	显示阵列内容	cellplot *	显示阵列的图形描述
	num2cell *	把数字数组变换为单元阵列	deal *	把输入分配给输出
	cell2struct *	把单元阵列变换为结构阵列	struct2cell	把结构阵列变换为单元阵列
	iscell *	是单元阵列时为真		
结构函数	struct *	建立或变换为结构阵列	fieldnames	获取结构域名
	getfield *	获取结构域的内容	setfield *	设定结构域的内容
	rmfield *	去除结构域	isfield *	若域在结构阵列中时为真
	isstruct *	是结构时为真		
面向对象编程函数	class *	建立对象或返回对象类	struct *	把对象变换为结构类
	methods *	显示类的方法名	isa *	若对象是给定类时为真
	isobject *	是对象时为真	inferiorto *	下级类关系
	superiorto *	上级类关系		
可扩展的运算符*	minus	扩展的 a-b	plus	扩展的 a+b
	times	扩展的 a.*b	mtimes	扩展的 a*b
	mldivide	扩展的 a\b	mrdivide	扩展的 a/b
	rdivide	扩展的 a./b	ldivide	扩展的 a.\b

续表

可扩展的运算符*	power	扩展的 a.^b	mpower	扩展的 a^b
	uminus	扩展的 -a	uplus	扩展的 +a
	horzcat	扩展的 [a b]	vertcat	扩展的 [a;b]
	le	扩展的 a<=b	lt	扩展的 a<b
	gt	扩展的 a>b	ge	扩展的 a>=b
	eq	扩展的 a==b	ne	扩展的 a~=b
	not	扩展的 ~a	and	扩展的 a&b
	or	扩展的 a|b	subsasgn	扩展的 a(i)=b,a{i}=b 和域=b
	subsref	扩展的 a(i), a{i}和一个域	colon	扩展的 a:b
	transpose	扩展的 a.'	ctranspose	扩展的 a'
	subsindex	扩展的 x(a)		

第 2 篇　应　用　篇

本篇把MATLAB语言用于解决电子信息领域的4门主要课程(即电路、信号与系统、数字信号处理及自动控制原理）中的问题。共包含近百个示例，基本覆盖了这些课程的主要题型。前两门课程（即第 5，6 两章）完全用 matlab 的基本函数编程，不用工具箱。其好处一是使读者真正从基本原理上理解课程；二是可以作为 MATLAB 编程技巧的训练。到第 7 章和第 8 章，才引入了信号处理和控制系统工具箱。为了节省篇幅，在书中所给的程序往往只列出了其主干部分，省略了绘图、标注等大同小异的大量语句。读者可以自行补编，也可购买本书的软盘，盘中的程序是完整的，并且是可以执行的。第 9 章是对 MATLAB 中其他工具箱的简单浏览，目的是扩展读者在应用 MATLAB 时的思路。

第5章　MATLAB在电路中的应用

MATLAB中的变量与常量都是矩阵（标量可看做1×1阶的矩阵，向量可看做n×1或1×n阶的矩阵），其元素可以是复数和任意形式的表达式，它具有元素群运算能力。MATLAB的这些优于其他语言的特色，有利于分析计算电路的各种问题，并且使编程更简便，运算效率更高。

本章主要介绍一些计算电路问题的编程方法和技巧，使读者逐步熟悉MATLAB语言的使用。至于有些例题的解法本身，不一定最佳。另外，还有Spice等软件是专门用来解电路问题的。使用MATLAB的好处是用同一种语言来解各门课程的问题，容易熟练，并且可以找到其共同点，甚至调用共同的子程序，给读者在学习中带来很大的便利。

5.1　电阻电路

【例5.1】 电阻电路的计算

如图5.1所示的电路，已知：R_1=2Ω，R_2=4Ω，R_3=12Ω，R_4=4Ω，R_5=12Ω，R_6=4Ω，R_7=2Ω。

（1）如u_s=10V，求i_3，u_4，u_7；

（2）如已知u_4=6V，求u_s，i_3，u_7。

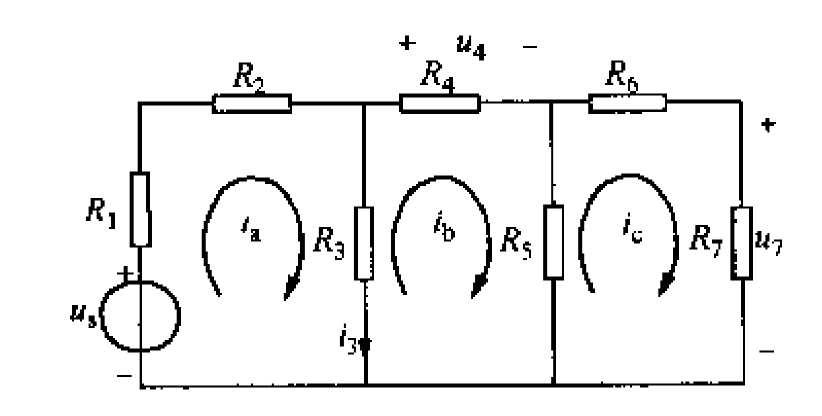


图5.1　例5.1的电路图

解：■ 建模

用网孔法，按图5.1可列得网孔方程为

$$(R_1+R_2+R_3)i_a-R_3i_b=u_s$$
$$-R_3i_a+(R_3+R_4+R_5)i_b-R_5i_c=0$$
$$-R_5i_b+(R_5+R_6+R_7)i_c=0$$

可写成如下所示的矩阵形式

$$\begin{bmatrix} R_1+R_2+R_3 & -R_3 & 0 \\ -R_3 & R_3+R_4+R_5 & -R_5 \\ 0 & -R_5 & R_5+R_6+R_7 \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}u_s$$

或直接列数字方程并简写为$\boldsymbol{AI}=\boldsymbol{B}u_s$

$$\begin{bmatrix} 2+4+12 & -12 & 0 \\ -12 & 12+4+12 & -12 \\ 0 & -12 & 12+4+2 \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}u_s$$

① 令u_s=10V，由$i_3=i_a-i_b$，$u_4=R_4i_b$，$u_7=R_7i_c$即可得到问题（1）的解。

② 由电路的线性性质，可令$i_3=k_1u_s$，$u_4=k_2u_s$，$u_7=k_3u_s$。

根据问题（1）的结果并根据如图5.1所示的电路可列出下式

$$k_1=\frac{i_3}{u_s}，k_2=\frac{u_4}{u_s}，k_3=\frac{u_7}{u_s}$$

于是，可以通过下列式子求得问题（2）的解

$$u_s=u_4/k_2 \qquad i_3=k_1u_s=\frac{k_1}{k_2}u_4$$

$$u_7=k_3u_s=\frac{k_3}{k_2}u_4$$

■ MATLAB 程序 q501.m

```
clear, close all, format compact
R1=2; R2=4; R3=12; R4=4; R5=12; R6=4; R7=2;          %为给定元件赋值
display('解问题（1）') % 解问题（1）
a11=R1+R2+R3; a12=-R3; a13=0;     %将系数矩阵各元素赋值
a21=-R3; a22=R3+R4+R5; a23=-R5;
a31=0; a32=-R5; a33=R5+R6+R7;
b1=1; b2=0; b3=0;
us=input('us='),                   % 输入解(1)的已知条件
A=[a11, a12, a13; a21, a22, a23; a31, a32, a33]             % 列出系数矩阵 A
B=[b1; 0; 0]; I=A\B*us;                              % I=[ia; ib; ic]
ia=I(1); ib=I(2); ic=I(3);
i3=ia-ib, u4=R4*ib, u7=R7*ic          % 解出所需变量
display('解问题（2）') % 利用电路的线性性质及问题（1）的解求解问题（2）
u42=input('给定 u42=');
k1=i3/us; k2=u4/us; k3=u7/us;          % 由问题（1）得出待求量与 us 的比例系数
us2=u42/k2, i32=k1/k2*u42, u72=k3/k2*u42   % 按比例方法求出所需变量
```

■ 程序运行结果

解问题（1）

给定 us=10

```
i3 = 0.3704,          u4 = 2.2222,        u7 = 0.7407
```

解问题（2）

给定 u42=6

```
us2 = 27.0000,          i32 =1.0000,        u72 = 2
```

答案

（1）　i_3=0.3704A，u_4=2.2222V，u_7=0.7404V

（2）　u_s=27V，i_3=1A，u_7=2V

实际上，如果熟悉列方程的方法，那么在编写 MATLAB 程序时可直接写出 A 和 B 为

```
A=[2+4+12  -12     0; -12    12+4+12    -12; 0      -12      12+4+2]
B=[1         0       0]
```

从而可省去给元件和矩阵各元素赋值等语句。

【例 5.2】 含受控源的电阻电路

如图 5.2 的电路，已知 $R_1=R_2=R_3=4\,\Omega$，$R_4=2\,\Omega$，控制常数 k_1=0.5，k_2=4，i_s=2A，求 i_1 和 i_2。

解：■ 建模

按图 5.2 列出节点方程：

$$\left(\frac{1}{R_1}+\frac{1}{R_2}\right)u_a-\frac{1}{R_2}u_b=i_s+k_1i_2$$

及

$$-\frac{1}{R_2}u_a+\left(\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4}\right)u_b=-k_1i_2+\frac{k_2i_1}{R_3}$$

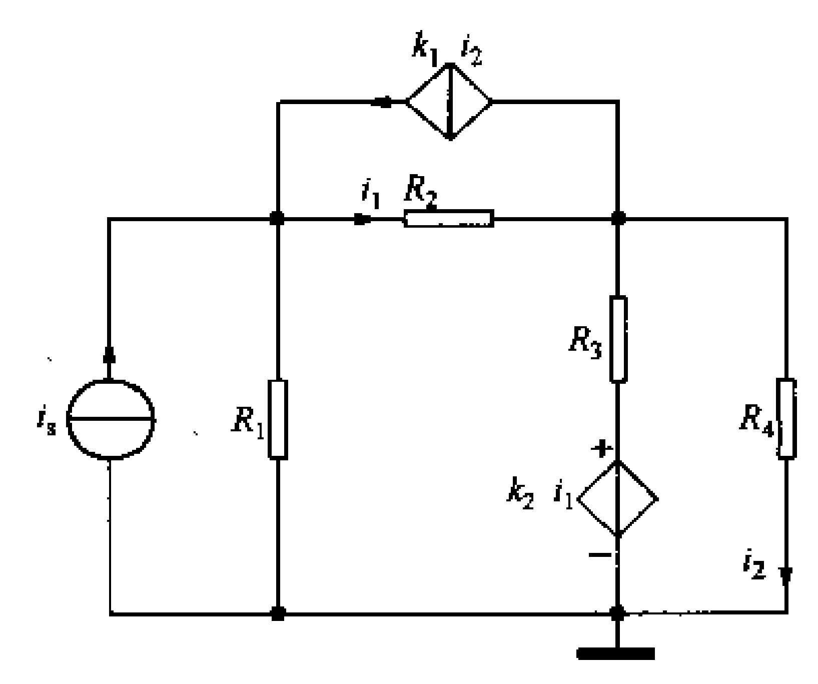


图 5.2　例 5.2 的电路图

由图 5.2 可知，控制变量 i_1，i_2 与节点电压 u_a，u_b 的关系为

$$i_1=\frac{u_a-u_b}{R_2},\quad i_2=\frac{u_b}{R_4}$$

整理以上各式，将 i_1，i_2 也作为未知量移至等号左端，并写成矩阵形式为

$$\begin{bmatrix}\frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} & 0 & -k_1\\ -\frac{1}{R_2} & \frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_4} & -\frac{k_2}{R_3} & k_1\\ \frac{1}{R_2} & -\frac{1}{R_2} & -1 & 0\\ 0 & \frac{1}{R_4} & 0 & -1\end{bmatrix}\begin{bmatrix}u_a\\ u_b\\ i_1\\ i_2\end{bmatrix}=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}i_s$$

令 i_s=2A，解上式即得 i_1 和 i_2。

■ MATLAB程序q502.m

```
clear, format compact
R1=4; R2=4; R3=4; R4=2;        % 设置元件参数
is=2; k1=0.5; k2=4;
% 按A*X=B*is列写此电路的矩阵方程，其中X=[ua; ub; i1; i2]。
a11=1/R1+1/R2; a12=-1/R2; a13=0; a14=-k1;                  % 设置系数A
a21=-1/R2; a22=1/R2+1/R3+1/R4; a23=-k2/R3; a24=k1;
a31=1/R2; a32=-1/R2; a33=-1; a34=0;
a41=0; a42=1/R4; a43=0; a44=-1;
A=[a11,a12,a13,a14; a21,a22,a23,a24; a31,a32,a33,a34; a41,a42,a43,a44];
B=[1; 0; 0; 0];                % 设置系数B
X=A\B*is;                                       % 解出X
i1=X(3), i2=X(4)             % 显示要求的分量
```

■ 程序运行结果

```
i1 = 1, i2 = 1
```

答案　i_1=1A，i_2=1A。

【例 5.3】　戴维南定理

如图 5.3-1（a）所示电路，已知：R_1=4Ω，R_2=2Ω，R_3=4Ω，R_4=8Ω；i_{s_1}=2A，i_{s_2}=0.5A

（1）负载 R_L 为何值时能获得最大功率？

（2）研究 R_L 在 0～10 Ω 范围内变化时，其吸收功率的情况。

解：■ 建模

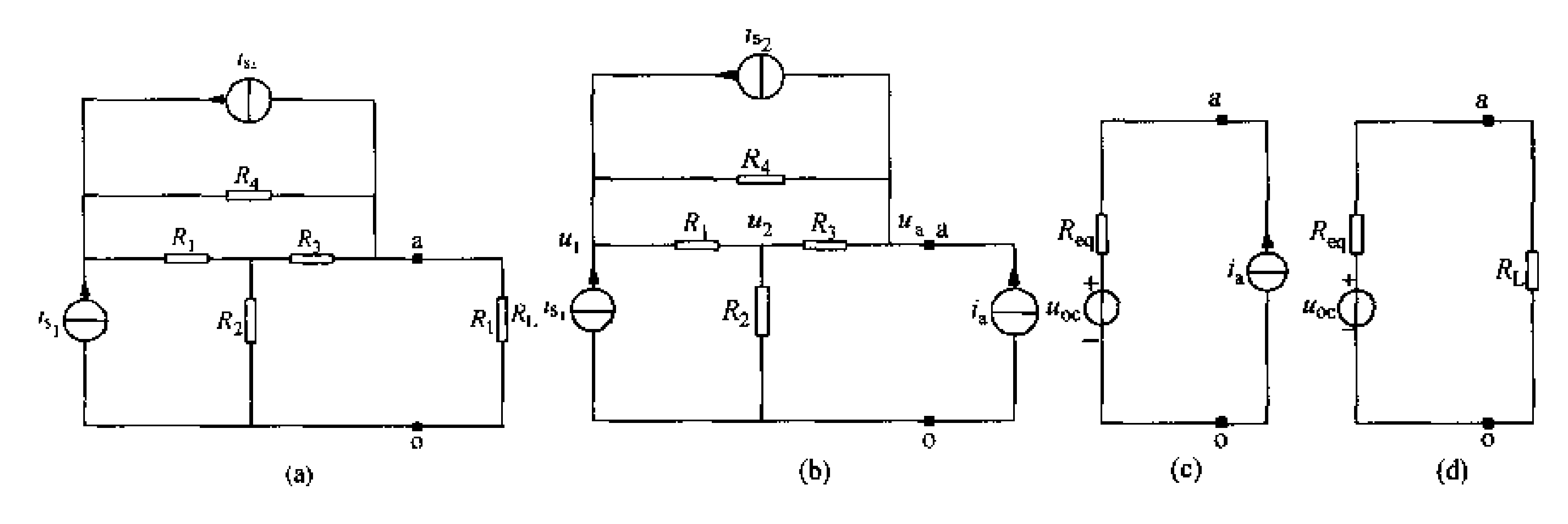


图 5.3-1　例 5.3 等效电路变换

先求图 5.3-1（a）中 ao 端以左的戴维南等效电路。为此断开 ao，并在 ao 端接入外电流源 i_a，如图 5.3-1（b）所示。取 o 为参考点，可列出节点方程

$$\begin{cases}\left(\dfrac{1}{R_1}+\dfrac{1}{R_4}\right)u_1-\dfrac{1}{R_1}u_2-\dfrac{1}{R_4}u_a=i_{s_1}+i_{s_2}\\ -\dfrac{1}{R_1}u_1+\left(\dfrac{1}{R_1}+\dfrac{1}{R_2}+\dfrac{1}{R_3}\right)u_2-\dfrac{1}{R_3}u_a=0\\ -\dfrac{1}{R_4}u_1-\dfrac{1}{R_3}u_2+\left(\dfrac{1}{R_3}+\dfrac{1}{R_4}\right)u_a=-i_{s_2}+i_a\end{cases}\tag{5.1}$$

写成矩阵为：

$$A\begin{bmatrix}u_1\\u_2\\u_a\end{bmatrix}=\begin{bmatrix}1&1&0\\0&0&0\\0&-1&1\end{bmatrix}\begin{bmatrix}i_{s_1}\\i_{s_2}\\i_a\end{bmatrix}\tag{5.2}$$

式中

$$A=\begin{bmatrix}\dfrac{1}{R_1}+\dfrac{1}{R_4} & -\dfrac{1}{R_1} & -\dfrac{1}{R_4}\\ -\dfrac{1}{R_1} & \dfrac{1}{R_1}+\dfrac{1}{R_2}+\dfrac{1}{R_3} & -\dfrac{1}{R_4}\\ -\dfrac{1}{R_4} & -\dfrac{1}{R_3} & \dfrac{1}{R_3}+\dfrac{1}{R_4}\end{bmatrix}$$

（注：这类数学表达式，由于线性代数中已学过，为节省篇幅，以后不再一一列举）

戴维南等效电路如图 5.3-1(c)，其方程为：

$$u_a=R_{eq}i_a+u_{oc}\tag{5.3}$$

● 方法 1

令 $i_a=0$，$i_{s_1}=2A$，$i_{s_2}=0.5A$，由式(5.2) 解得 u_{11},u_{21},u_{a1}。

因 $i_a=0$，由式(5.3)得　$u_{oc}=u_{a1}$。

再令 $i_{s_1}=i_{s_2}=0$，$i_a=1A$，仍由式(5.2) 解得另一组 u_{12},u_{22},u_{a2}。

因为内部电源 $i_{s_1}=i_{s_2}=0$，故 $u_{oc}=0$。由式(5.3)得

$$R_{eq}=\frac{u_{a2}}{i_a}=u_{a2}$$

于是可做出图 5.3-1（a）电路的戴维南等效电路，如图 5.3-1（d），R_L 获得最大功率时

$$R_L=R_{eq}$$

$$P_{Lmax}=\frac{u_{oc}^2}{4R_{eq}}$$

对于问题（2），由图 5.3-1（d）可得 R_L 吸收功率

$$P_L=\frac{R_L u_{oc}^2}{(R_{eq}+R_L)^2}$$

令 R_L=1Ω，2Ω，3Ω，…，10Ω，分别求得 P_L，并画图。

● 方法 2

设 i_a 为一个序列，（例如 i_a=0.1，0.2，…，2），计算相应的 u_a 序列，用线性拟合，得出直线方程（5.3），即

$$u_a=c(2)i_a+c(1)$$

从而求得　$u_{oc}=c(1)$，$R_{eq}=c(2)$。

■ **MATLAB程序q503.m**

```
clear, format compact
R1=4; R2=2; R3=4; R4=8;            % 设置元件参数
is1=2; is2=0.5;
% 按 A*X=B*is 列写此电路的矩阵方程，其中 X=[u1; u2; ua]; is=[is1; is2; ia]
a11=1/R1+1/R4; a12=-1/R1; a13=-1/R4;                 % 设置系数矩阵 A
a21=-1/R1; a22=1/R1+1/R2+1/R3; a23=-1/R3;
a31=-1/R4; a32=-1/R3; a33=1/R3+1/R4;
A=[a11, a12, a13; a21, a22, a23; a31, a32, a33];
B = [1, 1, 0; 0, 0, 0; 0, -1, 1];                       % 设置系数矩阵 B
% 方法 1: 令 ia=0, 求 uoc=x1(3); 再令 is1=is2=0, 设 ia=1, 求 Req=ua/ia=x2(3).
X1=A\B*[is1; is2; 0]; uoc=X1(3)
X2=A\B*[0; 0; 1]; Req=X2(3)
RL=Req; P=uoc^2*RL/(Req+RL)^2                  % 求最大负载功率
% 也可设 RL 为一数组，求出的负载功率也为一数组，画出曲线找极大值
RL=0:10, p=(RL*uoc./(Req+RL)).*uoc./(Req+RL), % 设 RL 序列，求其功率
figure(1), plot(RL, p), grid         % 画出功耗随 RL 变化的曲线如图 5.3-2(a)
% 方法 2: 设一个 ia 序列，计算一个 ua 序列，用线性拟合求出其等效开路电压和等效内阻
% for k=1:21
%     ia(k)=(k-1)*0.1;
%     X=A\B*[is1; is2; ia(k)];                      % 定义 X=[u1; u2; ua]
%     u(k)=X(3); end
```

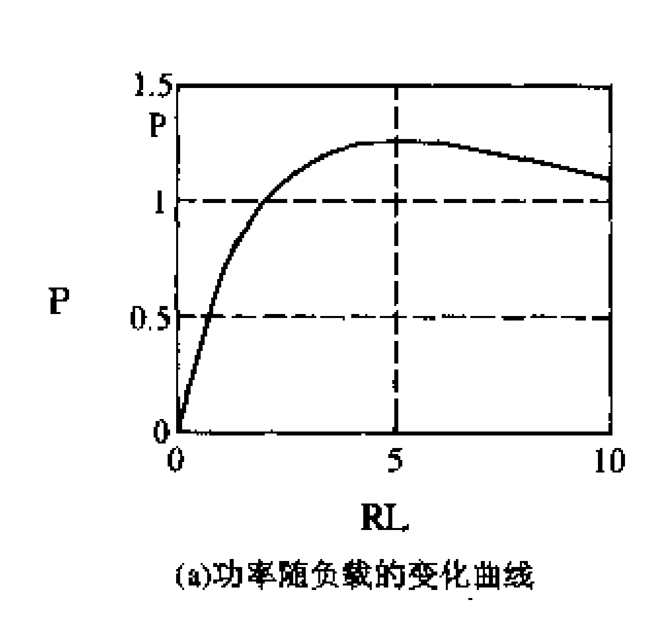
```
% figure(2), plot(ia, u, 'x'), grid   % 线性拟合，见图 5.3-2(b)
% c=polyfit(ia, u, 1)   ;           % ua=c(2)*ia+c(1), 用拟合函数求 c(1), c(2)
% uoc=c(1), Req=c(2)
```

■ **程序运行结果**

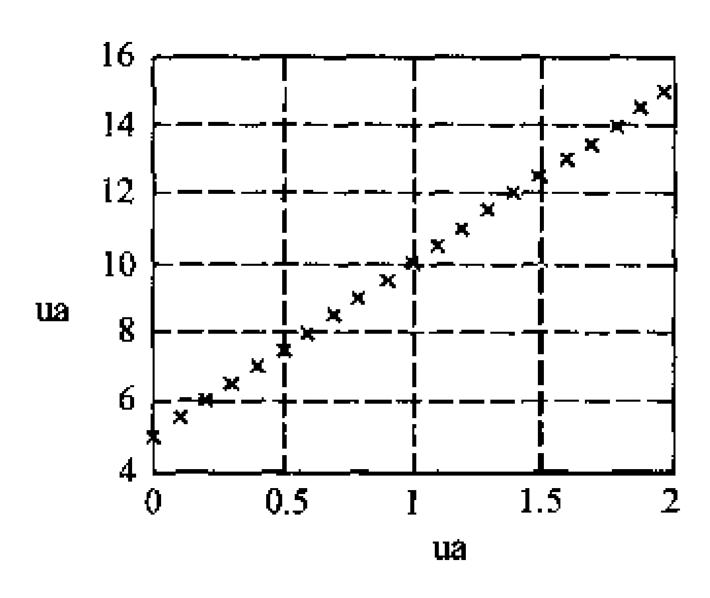
Uoc=5V,

Req=5Ω,

Pmax=1.25W。



(a)功率随负载的变化曲线



(b) 电路对负载的输出特性

图 5.3-2　例 5.3 的解

5.2 动 态 电 路

【例 5.4】 一阶动态电路，三要素公式

如图 5.4-1 所示电路，已知：R_1=3Ω，R_2=12Ω，R_3=6Ω，C=1F；u_s=18V，i_s=3A，在 t<0 时，开关 S 位于"1"，电路已处于稳定状态。

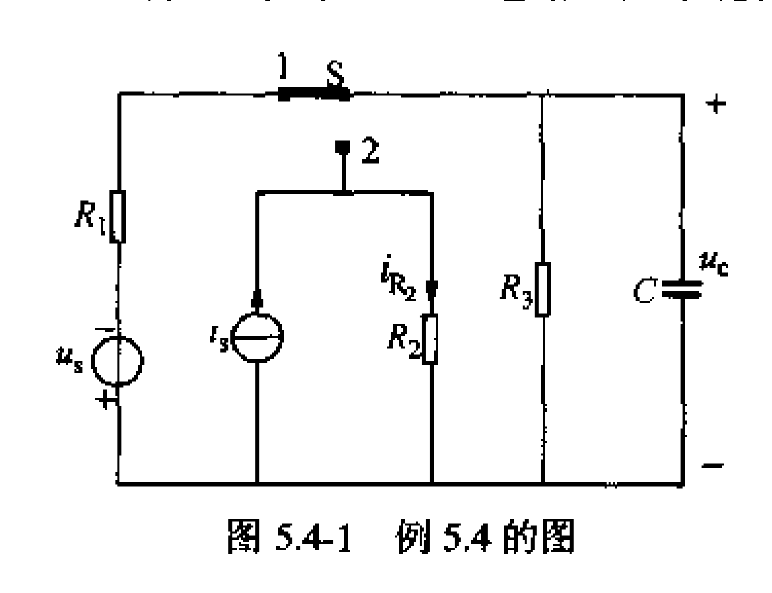


图 5.4-1　例 5.4 的图

（1）t=0 时，开关 S 闭合到"2"，求 $u_c(t)$，$i_{R_2}(t)$，并画出波形；

（2）若经 10 秒，开关 S 又复位到"1"，求 $u_c(t)$，$i_{R_2}(t)$，并画出波形。

解：■ 建模

这是一阶动态电路，可用三要素公式求解。

（1）首先求初始值 $u_c(0_+)$和 $i_{R_2}(0_+)$，为此先求 $u_c(0_-)$，在 $t=0_-$时，开关位于"1"，电路已达到稳定。电容可看做开路，不难求得 $u_c(0_-)=-12$V。

根据换路时电容电压不变的定律，得电容初始电压

$$u_c(0_+)=u_c(0_-)=-12\text{V}$$

在 t=0 时，开关已闭合到"2"，可求得非独立初始值

$$i_{R_2}(0_+)=\frac{u_c(0_+)}{R_2}=-1\text{A}$$

其次求稳定值。达到稳态时电容可看做开路，于是可得

$$u_c(\infty)=\frac{R_2R_3}{R_2+R_3}i_s$$

$$i_{R_2}(\infty)=\frac{R_3}{R_2+R_3}i_s$$

时常数
$$\tau_1=\frac{R_2R_3}{R_2+R_3}C$$

用三要素公式得

$$u_c(t)=u_c(\infty)+[u_c(0_+)-u_c(\infty)]e^{-t/\tau_1} \qquad t\geqslant 0 \qquad (5.4)$$

$$i_{R_2}(t)=i_{R_2}(\infty)+[i_{R_2}(0_+)-i_{R_2}(\infty)]e^{-t/\tau_1} \qquad t\geqslant 0 \qquad (5.5)$$

（2）经 10 秒后，开关又闭合到“1”，将 t=10s 代入（5.4）即得电容电压的初始值为 $u_c(10_+)=u_c(10)$。由图 5.4-1 可见这时 $i_{R_2}(10_+)=i_s$=3A，并保持不变。达到稳定时，$u_c'(\infty)=-12$V

这时
$$\tau_2=\frac{R_1R_3}{R_1+R_3}C,$$

用三要素公式得

$$u_c(t)=\begin{cases}12-24e^{-t/\tau_1}, & 0\leqslant t\leqslant 10\\ u_c'(\infty)+[u_c(10_+)-u_c'(\infty)]e^{-\frac{t-10}{\tau_2}}, & 10\leqslant t\end{cases}$$

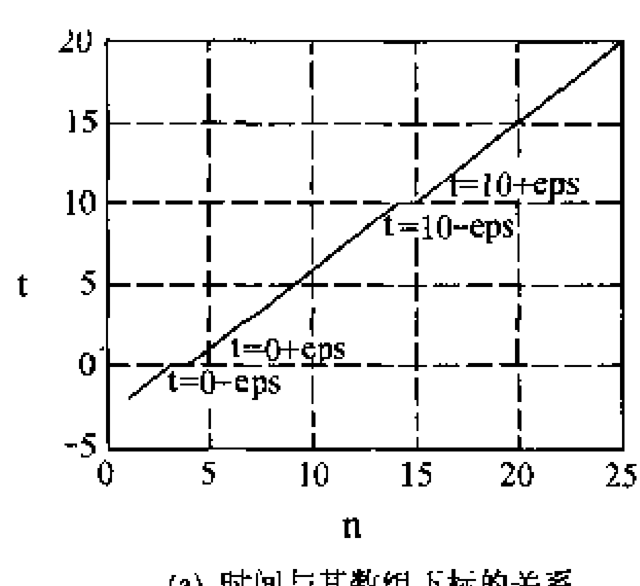
$$i_{R_2}(t)=\begin{cases}3 & t<0\\ 1-2e^{t/\tau_1} & 0\leqslant t\leqslant 10\\ 3 & t>10\end{cases}$$

■ MATLAB程序q504.m

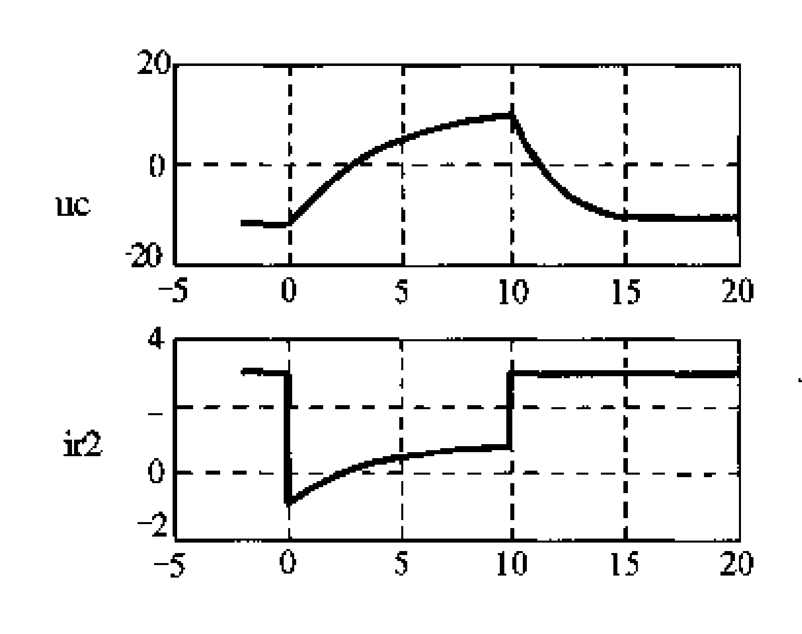
```
R1=3; us=18; is=3; R2=12; R3=6; C=1;  % 给出原始数据
% 解问题(1)
uc0=-12; ir20=uc0/R2; ir30=uc0/R3;    % 算出初值 ir20 及 uc0
ic0=is-ir20-ir30;
ir2f=is*R3/(R2+R3);                 % 算出终值 ir2f 及 ucf
ir3f=is*R2/(R2+R3);
ucf=ir2f*R2;  icf=0;
% 注意时间数组的设置，在 t=0 及 10 附近设两个点，见图 5.4-2(a)
t=[-2-eps:0-eps, 0:9, 10-eps, 10+eps, 11:20];
figure(1),plot(t),grid
% 从图 5.4B(a)中可看出时间与时间数组下标的关系，t=10+eps 对应下标 15
uc(1:3)=-12; ir2(1:3)=3;            % t<0 时的值
T = R2*R3/(R2+R3)*C;                % 求充电时常数
uc(4:14)=ucf+(uc0-ucf)*exp(-t(4:14)/T);     %
ir2(4:14)=ir2f+(ir20-ir2f)*exp(-t(4:14)/T);    % 用三要素法求输出
% 解问题(2)
uc(15)=uc(14); ir2(15)=is;                 % 求 t=10+eps 时的各初值
ucf2=-12; ir2f=is;                         % 求 uc 和 ir2 在新区间终值 ucf2 和 ir2f
```

```
T2=R1*R3/(R1+R3)*C;                    % t=10+eps 到 t=20 区间的时常数
% 再用三要素法求输出
uc(15:25)=ucf2+(uc(15)-ucf2)*exp(-(t(15:25)-t(15))/T2);
ir2(15:25)=is;
figure(2)
subplot(2, 1, 1); h1=plot(t, uc);      % 绘 uc 图
grid, set(h1, 'linewidth', 2)                    % 加大线宽
subplot(2, 1, 2), h2=plot(t, ir2);     % 绘 ir2 图
grid, set(h2, 'linewidth', 2)
```

■ **程序运行结果**



(a) 时间与其数组下标的关系



(b) u_c 及 i_{R_2} 的暂态波形

图 5.4-2　例 5.4 的解

【例 5.5】　正弦激励的一阶电路

如图 5.5-1 的一阶电路，已知 $R=2\Omega$，$C=0.5F$，电容初始电压 $u_c(0_+)=4V$，激励的正弦电压 $u_s(t)=u_m\cos\omega t$，其中 $u_m=10V$，$\omega=2rad/s$。当 $t=0$ 时，开关 S 闭合，求电容电压的全响应，区分其暂态响应与稳态响应，并画出波形。

解：■ 建模

图 5.5-1 中电容电压的微分方程为

$$\frac{du_c}{dt}+\frac{1}{RC}u_c=\frac{1}{RC}u_s$$

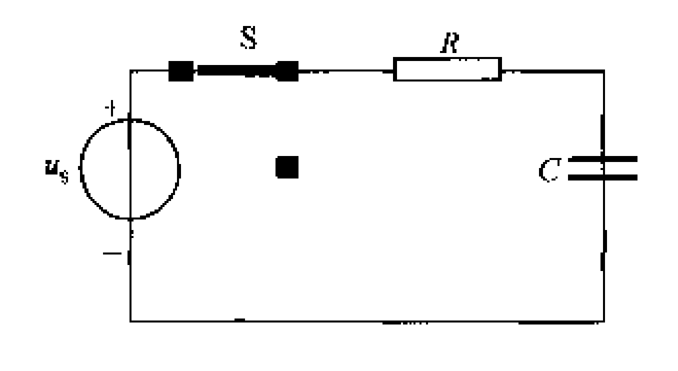


图 5.5-1　例 5.5 电路图

其时间常数　$\tau=RC$，

若用三要素公式，其解为

$$u_c(t)=u_{cp}(t)+[u_c(0_+)-u_{cp}(0_+)]e^{-t/\tau},\quad t\geqslant 0$$

式中 $u_c(0_+)$为电容初始电压，$u_{cp}(t)$为微分方程的特解，当正弦激励时，设 $u_{cp}(t)=u_{cm}\cos(\omega t+\varphi)$

式中

$$u_{cm}=\frac{\frac{1}{\omega C}u_m}{\sqrt{R^2+\left(\frac{1}{\omega C}\right)^2}}$$

$$\varphi = 90^\circ - \arctan\frac{1}{\omega CR}$$

$$u_{cp}(0_+)=u_{cm}\cos\varphi$$

最后得电容电压的全响应　　$u_c(t)=[u_c(0_+)-u_{cp}(0_+)]e^{-t/\tau}+u_{cm}\cos(\omega t+\varphi)$

其暂态响应（固有响应）　　$u_{ctr}(t)=[u_c(0_+)-u_{cp}(0_+)]e^{-t/\tau}$　　$t\geqslant 0$

稳态响应（强迫响应）　　$u_{ctr}(t)=u_{cm}\cos(\omega t+\varphi)$

■ MATLAB程序q505.m

```
R=2;  C=0.5;  T=R*C;  uc0=4; % 输入元件参数
um=10;  w=2;  Zc=1/j/w/C;
%输入给定参数
t=0:0.1:10;          % 设定时间数组
us=um*cos(w*t);      % 设定激励信号
ucst=us*Zc/(R+Zc);     % 稳态分量计算
ucp0=ucst(1);        % 稳态分量的初始值
uctr=(uc0-ucp0)*exp(-t/T); %暂态分量
uc=uctr+ucst; % 总的 uc 为两项之和
%把三种数据画在一张图上
plot(t, uc, '-', t, uctr, ':', t, ucst, '-.'),grid
legend('uc', 'uctr', 'ucst')% 用图例标注
```

■ 程序运行结果

得出电容上的暂态、稳态和总电压波形曲线如图 5.5-2。

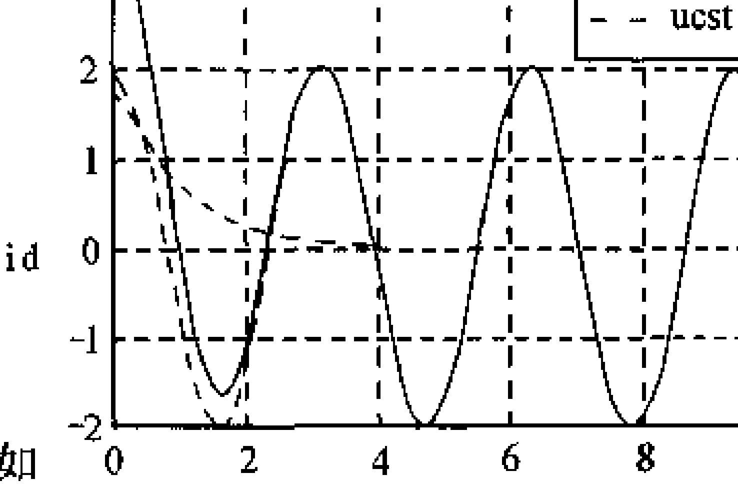


图 5.5-2　电容上的电压波形

【例 5.6】 二阶过阻尼电路的零输入响应

图 5.6-1 是典型的二阶动态电路，其零输入响应有过阻尼、临界阻尼和欠阻尼三种情况。本例讨论过阻尼情况，（临界阻尼和欠阻尼见例 5.7）。如已知 L=0.5H，C=0.02F，R=12.5Ω，初始值 $u_c(0)$=1V，$i_L(0)$=0，求 $t\geqslant 0$ 时的 $u_c(t)$和 $i_L(t)$的零输入响应，并画出波形。

解：■ 建模

● 方法 1

按图 5.6-1，不难列出，u_c的微分方程（i_L的方程类似，略）为

$$\frac{d^2u_c(t)}{dt^2}+\frac{R}{L}\frac{du_c(t)}{dt}+\frac{1}{LC}u_c(t)=0$$

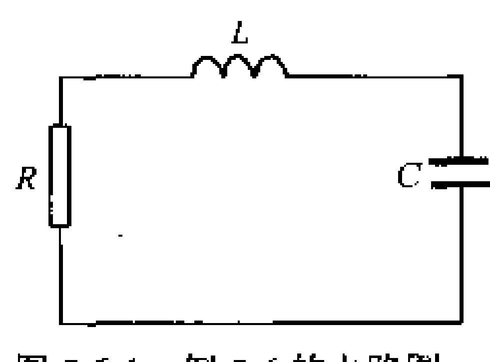


图 5.6-1　例 5.6 的电路图

令衰减常数 $\alpha=\frac{R}{2L}$，谐振角频率 $\omega_n=\frac{1}{\sqrt{LC}}$，则上式可写为二阶微分方程的典型形式

$$\frac{d^2u_c}{dt^2}+2\alpha\frac{du_c}{dt}+\omega_n^2u_c=0$$

其初始值为　　$u_c(0)$ 和 $\frac{du_c}{dt}\Big|_{t=0}=\frac{i_L(0)}{C}$

本例中

$$\alpha=\frac{R}{2L}=12.5\ ,\omega_n=\frac{1}{\sqrt{LC}}=10$$

即有$\alpha>\omega_n$的过阻尼情况。其解为

$$u_c(t)=\frac{p_2u_c(0)-\dfrac{i_L(0)}{C}}{p_2-p_1}e^{p_1t}-\frac{p_1u_c(0)-\dfrac{i_L(0)}{C}}{p_2-p_1}e^{p_2t}$$

$$i_L(t)=\frac{p_1C\left[p_2u_c(0)-\dfrac{i_L(0)}{C}\right]}{p_2-p_1}e^{p_1t}-\frac{p_2C\left[p_1u_c(0)-\dfrac{i_L(0)}{C}\right]}{p_2-p_1}e^{p_2t}$$

式中有　　$p_1=-\alpha+\sqrt{\alpha^2-\omega_n^2}$　，　$p_2=-\alpha-\sqrt{\alpha^2-\omega_n^2}$

● 方法 2

对微分方程作拉普拉斯变换，考虑到初始条件，可得

$$s^2U_c(s)-su_c(0_)-\frac{du_c}{dt}(0_)+2\alpha[sU_c(s)-u_c(0_)]+\omega_n{}^2U_c(s)=0$$

整理后得

$$U_c(s)=\frac{su_c(0_)+2\alpha u_c(0_)+i_L(0_)/C}{s^2+2\alpha s+\omega_n{}^2}$$

对它求拉普拉斯反变换，就可得到时域函数。为此可将等式右端的多项式分解为部分分式，得

$$U_c(s)=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}$$

其中，p_1和p_2是多项式分式的极点，而r_1和r_2则是它们对应的留数。便有

$$u_c(t)=r_1e^{p_1t}+r_2e^{p_2t}$$

p_1，p_2，r_1和r_2可以用代数方法求出，在MATLAB中有residue函数，专门用来求多项式分式的极点和留数，其格式为：

➢ [r，p，k]=residue(num，den)

其中num，den分别为分子、分母多项式系数组成的数组。进而写出：

u=r(1)*exp(p(1)*t)+ r(2)*exp(p(2)*t)+…

这样就无需求出其显式，程序特别简明。

■ MATLAB程序q506.m

```
clear,  format compact
L=0.5;  R=12.5;  C=0.02;            % 输入元件参数
uc0=1;  iL0=0;
alpha=R/2/L;  wn=sqrt(1/(L*C));  % 输入给定参数
p1=-alpha+sqrt(alpha^2-wn^2)              % 方程的两个根
p2=-alpha-sqrt(alpha^2-wn^2)
dt=0.01;  t=0:dt:1;                  % 设定时间数组
% 方法 1，用公式
uc1=(p2*uc0-iL0/C)/(p2-p1)*exp(p1*t);              % uc 的第一个分量
uc2=-(p1*uc0-iL0/C)/(p2-p1)*exp(p2*t);              % uc 的第二个分量
```

```
iL1=p1*C*(p2*uc0-iL0/C)/(p2-p1)*exp(p1*t);          % iL 的第一个分量
iL2=-p2*C*(p1*uc0-iL0/C)/(p2-p1)*exp(p2*t);         % iL 的第二个分量
uc=uc1+uc2;  iL=iL1+iL2;                            % 把两个分量相加
% 分别画出两种数据曲线
subplot(2, 1, 1), plot(t, uc), grid
subplot(2, 1, 2), plot(t, iL), grid
% 方法 2, 用拉普拉斯变换及留数法
num=[uc0, R/L*uc0+iL0/C];            % uc(s)的分子系数多项式
den=[1, R/L, 1/L/C];                 % uc(s)的分母系数多项式
[r, p, k]=residue(num, den);         % 求极点留数
% 求时域函数 ucn, 对 ucn 求导得到电流 iLn
ucn=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t);
iLn=C*diff(ucn)/dt;      %
% 绘曲线，注意求导后数据长度减少一个。
figure(2), subplot(2, 1, 1),
plot(t, ucn), grid % 绘曲线
subplot(2, 1, 2)
plot(t(1:end-1), iLn), grid
```

■ **程序运行结果**

两种方法的曲线形状相同，如图 5.6-2。

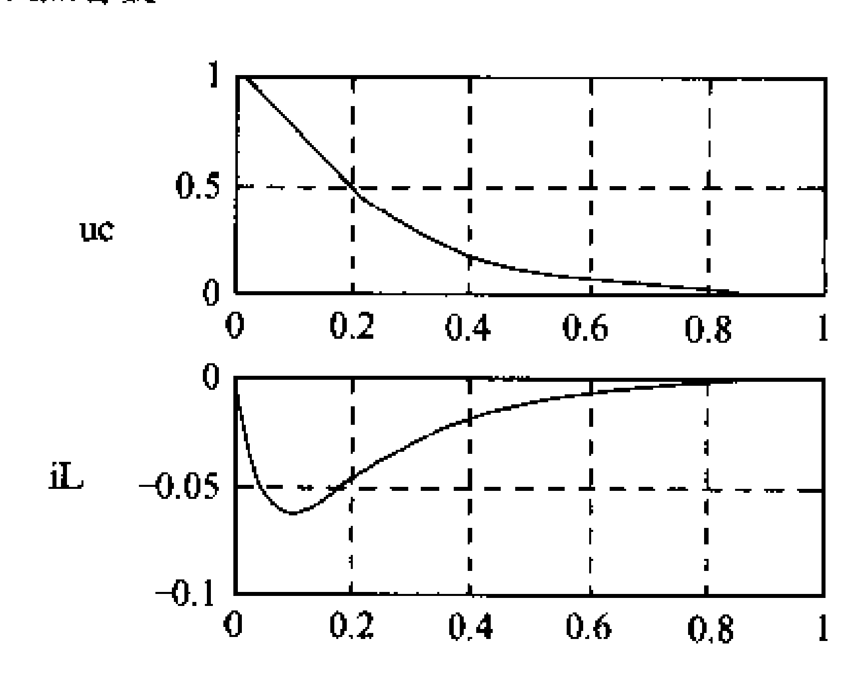


图 5.6-2　电压 u_c 和 i_L 电流波形

【例 5.7】 二阶欠阻尼电路的零输入响应

如图 5.6-1 的二阶电路，如 L=0.5H，C=0.02F。初始值 $u_c(0)$=1V，i_L=0，试研究 R 分别为 1Ω，2Ω，3Ω，…，10Ω时，$u_c(t)$和 $i_L(t)$的零输入响应，并画出波形图。

解：■ 建模

本例电路的微分方程同例 5.6，不再重复。这里 $\omega=\dfrac{1}{\sqrt{LC}}=10$，当 R=1Ω，2Ω，3Ω，…，10Ω时，α =1，2，3，…，10。显然$\alpha=\omega_n$=10 为临界阻尼，其余为欠阻尼（衰减振荡）情况，这时方程的解为

$$u_c(t)=Ae^{-\alpha t}\sin(\beta t+\varphi)$$

$$i_L(t)=-t\omega_n CAe^{\alpha t}\sin(\beta t-\theta)$$

式中

$$A=\sqrt{\frac{[\beta u_c(0)]^2+\left[\dfrac{i_L(0)}{C}+\alpha u_c(0)^2\right]}{\beta^2}}$$

$$\varphi=\arctan\frac{\beta\, u_c(0)}{\dfrac{i_L(0)}{C}+\alpha\, u_c(0)}\qquad \theta=\arctan\frac{\beta\dfrac{i_L(0)}{C}}{\alpha\dfrac{i_L(0)}{C}+\omega_n^2 u_c(0)}$$

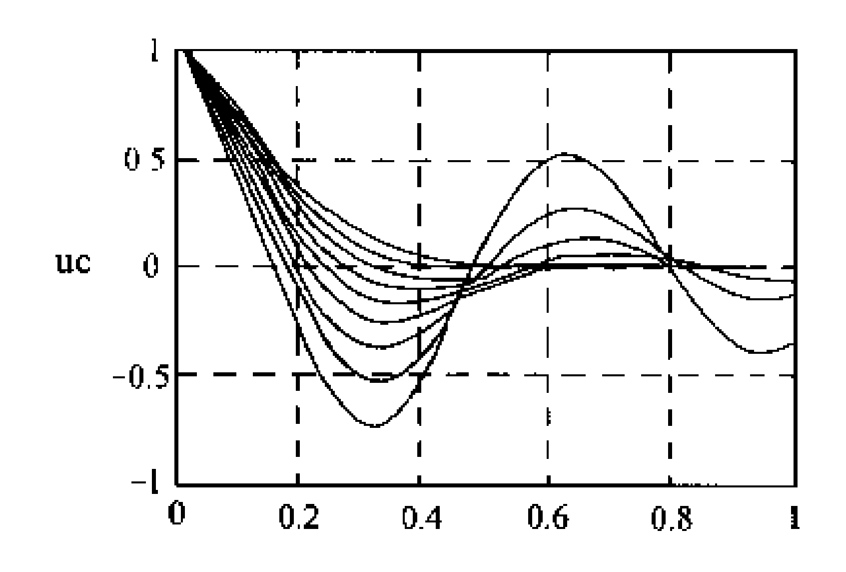
式中各公共参数 $\alpha=\dfrac{R}{2L}$，$\omega_n=\dfrac{1}{\sqrt{LC}}$，$\beta=\sqrt{\omega_n^2-\alpha^2}$ 。

■ MATLAB程序q507.m

```
clear, format compact
L=0.5; C=0.02;            % 输入元件参数
uc0=1; iL0=0;
for R=1:10
   alpha=R/2/L; wn=sqrt(1/(L*C));  % 输入给定参数
   p1=-alpha-sqrt(alpha^2-wn^2);       % 方程的两个根
   p2=-alpha+sqrt(alpha^2-wn^2);
   dt=0.01; t=0:dt:1;                  % 设定时间数组
   % 方法 1, 用公式
         beta=sqrt(wn^2-alpha^2);
         A=sqrt((beta*uc0)^2+(iL0/C+alpha*uc0)^2)/beta;
         phi=atan(beta*uc0/(iL0/C+alpha*uc0));
         theta=atan(beta*iL0/C/(alpha*iL0/C+wn^2*uc0));
         uc=A*exp(-alpha*t).*sin(beta*t+phi);
         iL=-wn*C*exp(-alpha*t).*sin(beta*t+theta);
   % 分别画出两种数据曲线
   figure(1), plot(t, uc), hold on
   figure(2), plot(t, iL), hold on
end
figure(1), grid, figure(2), grid
% 方法 2, 用拉普拉斯变换及留数法
   num=[uc0, R/L*uc0+iL0/C];                  % uc(s)的分子系数多项式
   den=[1, R/L, 1/L/C];            % uc(s)的分母系数多项式
   [r, p, k]=residue(num, den);    % 求极点留数
   ucn=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t);  % 求时域函数
   iLn=C*diff(ucn)/dt;             % 对 ucn 求导得到电流 iLn
   figure(1), plot(t, ucn), hold on        % 绘曲线
   figure(2), plot(t(2:end), iLn), hold on
```

■ 程序运行结果

设 R 为 1～10Ω，可以得出图 5.7 所示的曲线族。用方法 2 时，应将该段程序放到 for 循环内部，两种方法所得曲线形状相同，因此只画出一组。只有当 R=10Ω时，用方法 2 得出的结果有很大的误差，这是因为 residue 程序在遇到重根时会出现奇异解，导致结果不正确。



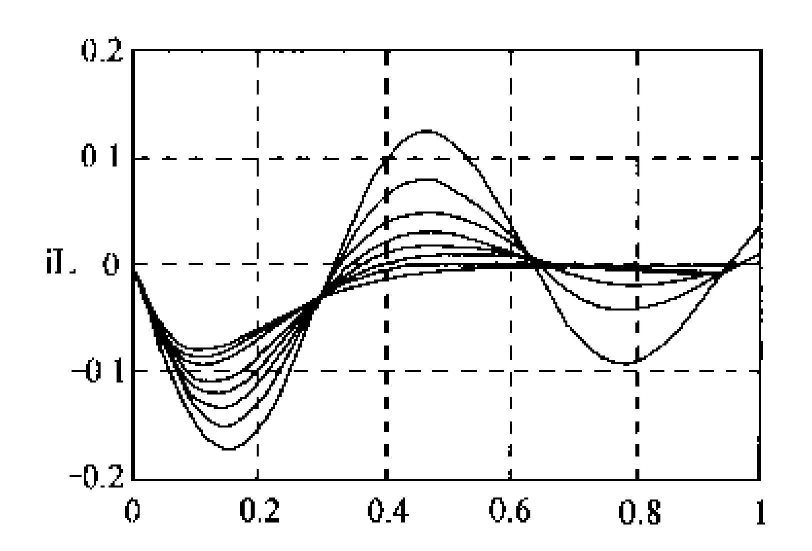


图 5.7　欠阻尼电路的零输入响应曲线与阻尼系数的关系

5.3　正弦稳态电路

【例 5.8】 简单正弦稳态电路

如图 5.8-1 所示电路，已知 R=5Ω，ωL=3Ω，$1/\omega C$=2Ω，U_c=10∠30° V，求 I_r，I_c，I，和 U_L，U_s。并画出其相量图（注：由于在 MATLAB 程序中，变量都看做复数，并且不能在变量上方加点。所以本书在文本和图中都省略了电相量上的点）。

解：■ 建模

这是普通的正弦稳态电路问题，设 Z_1=jωL，Z_2=R，Z_3=1/jωC，R 与 C 并联后的阻抗为 $Z_{23}=\dfrac{Z_3Z_2}{Z_2+Z_3}$，总阻抗为 Z=Z_1+Z_{23}。

由图可得

$$\dot I_r=\dot U_c/Z_2,\ \dot I_c=\dot U_c/Z_3, \dot I=\dot I_r+\dot I_c$$
$$\dot U_L=Z_1\dot I,\ \dot U_s=Z\dot I$$

I_r 及 I_c 可由 U_c 分别除以 Z_2 及 Z_3 得到 I_r= U_c/Z_2 及 I_c=U_c/Z_3

■ MATLAB程序q508.m

```
%注意其复数运算的优势
Z1=3*j; Z2=5; Z3=-2j; Uc=10*exp(30j*pi/180);          % 给定参数几输入
Z23=Z2*Z3/(Z2+Z3); Z=Z1+Z23;                                % 列写交流电路方程
Ic=Uc/Z3, Ir=Uc/Z2, I=Ic+Ir, U1=I*Z1, Us=I*Z
disp(' Uc      Ir        Ic         I          U1          Us  ')
disp('幅值'), disp(abs([Uc, Ir, Ic, I, U1, Us]))
disp('相角'), disp(angle([Uc, Ir, Ic, I, U1, Us])*180/pi)
% compass 是 MATLAB 中绘制复数相量图的命令，用它画相量图特别方便。
ha=compass([Uc, Ir, Ic, I, U1, Us, Uc]);
set(ha, 'linewidth', 3)
```

■ 程序运行结果

	Uc	Ir	Ic	I	U1	Us
幅值	10.0000	2.0000	5.0000	5.3852	16.1555	7.8102
相角	30.0000	30.0000	120.0000	98.1986	-171.8014	159.8056

画出的相量图见图 5.8-2。

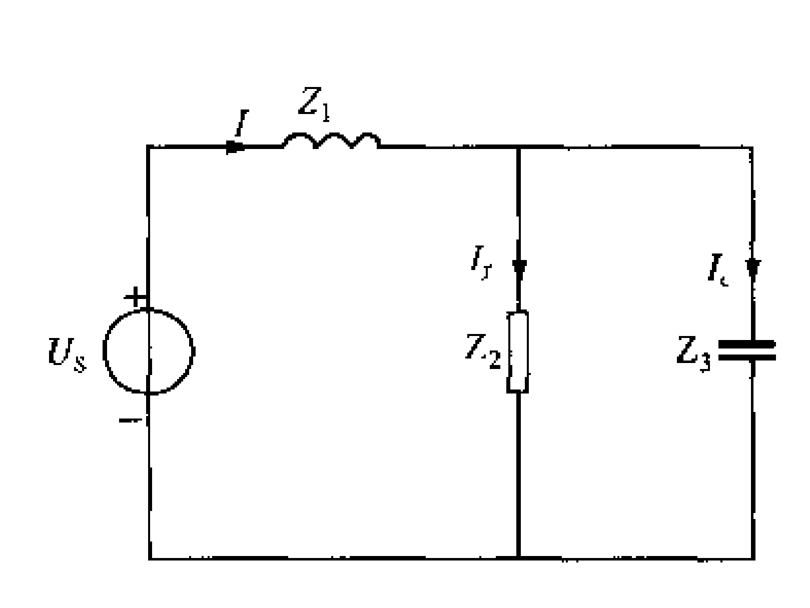



图 5.8-1　例 5.8 电路图

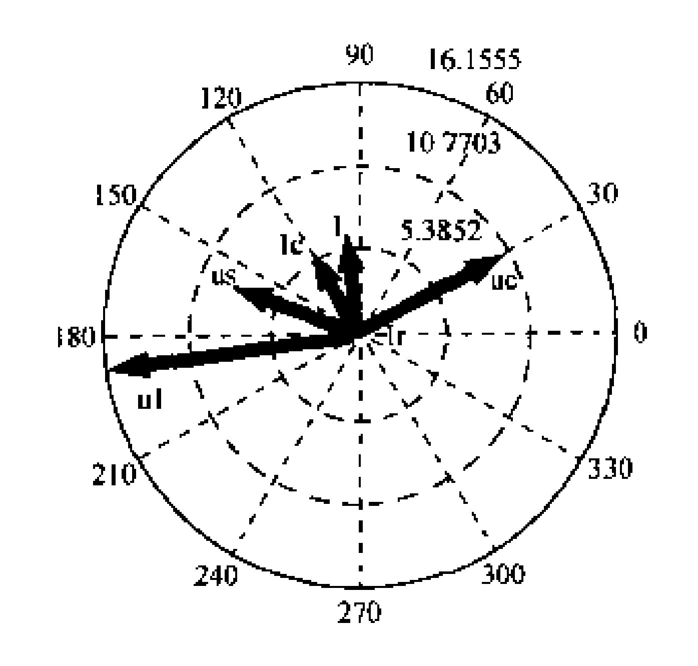


图 5.8-2　例 5.8 所得的相量图

通常给题时都设已知信号的初相角为零，本题故意将它设为 30°，此时的 U_c 为复数。将它写为指数形式为 $U_{cm}\exp(j\varphi)$，如程序中所示。

【例 5.9】 正弦稳态电路：戴维南定理

如图 5.9 所示电路，已知 C_1=0.5F，R_2=R_3=2Ω，L_4=1H；

U_s=10+10cos$t$，$I_s(t)$=5+5 cos2t，求 b，d 两点之间的电压 $U(t)$。

解：■ 建模

这是一个含三个频率分量的正弦稳态电路问题，可以按每个频率成分分别计算，再叠加起来。但是，更高明的办法是利用 MATLAB 的元素群计算特性，把多个频率分量及相应的电压、电流、阻抗等都看做多元素的行数组，每一元素对应于一种频率分量的值。因为他们服从同样的方程，所以程序就特别简洁。

（1）先看 $\dot{U}_s$ 对 b、d 点产生的等效电压 $\dot{U}_{oc}$，令电流源开路，即 $I_s=0$，由电桥电路可得

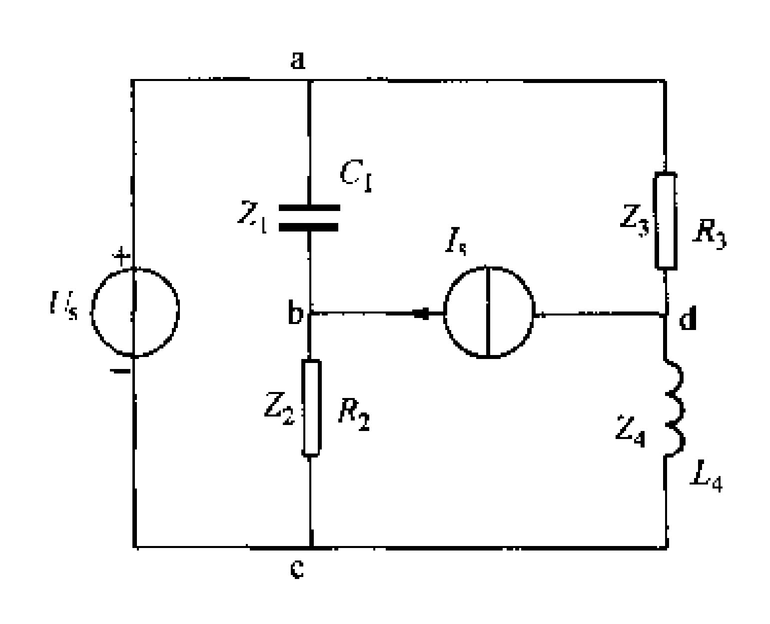


图 5.9　例 5.9 的电路图

$$\dot{U}_{oc}=\left[\frac{Z_2}{Z_1+Z_2}-\frac{Z_4}{Z_3+Z_4}\right]\dot{U}_s$$

其相应的等效内阻抗为

$$Z_{eq}=\frac{Z_3Z_4}{Z_3+Z_4}+\frac{Z_1Z_2}{Z_1+Z_2}$$

（2）令 $\dot{U}_s=0$，则电流源在 b，d 间产生的电压为 I_sZ_{eq}

（3）根据叠加原理

$$\dot{U}=\dot{I}_sZ_{eq}+\dot{U}_{oc}$$

■ MATLAB 程序 q509.m

```
clear, format compact
w=[eps, 1, 2]; Us=[10, 10, 0]; Is=[5, 0, 5]; % 按频率依次设定输入信号数组
Z1=1./(0.5*w*j); Z4=1*w*j;
% 电抗分量是频率的函数，故自动成为数组
Z2=[2, 2, 2]; Z3=[2, 2, 2];          % 对电阻分量也列写成常数数组
```

```
Uoc=(Z2./(Z1+Z2)-Z4./(Z3+Z4)).*Us;          % 列出电路的复数方程
Zeq=Z3.*Z4./(Z3+Z4)+Z1.*Z2./(Z1+Z2);   % 列出等效阻抗
U=Is.*Zeq+Uoc;                                     % 求解
disp('    w        Um        phi   ')           % 显示
disp([w', abs(U'), angle(U')*180/pi])
```

■ 程序运行结果

```
     w         Um           phi
   0.0000    10.0000         0
   1.0000     3.1623      -18.4349
   2.0000     7.0711       -8.1301
```

由此可以写出 $U(t)$的表示式：

$$U(t)=10+3.1623\cos(t-18.4349)+7.0711\cos(2t-8.1301)$$

【例 5.10】 含受控源的电路：戴维南定理

如图 5.10-1 所示电路，设 $Z_1=-j250\,\Omega$， $Z_2=250\Omega$，$I_s=2\angle 0°$ A，求负载 Z_L 获得最大功率时的阻抗值及其吸收功率。

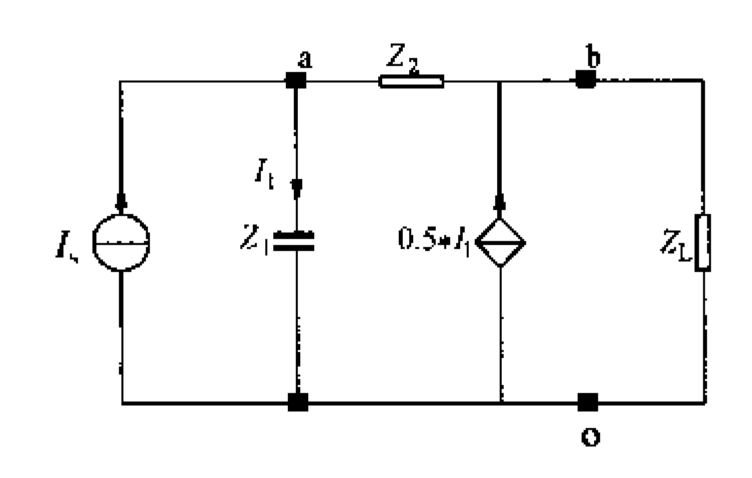


图 5.10-1　求戴维南等效电路

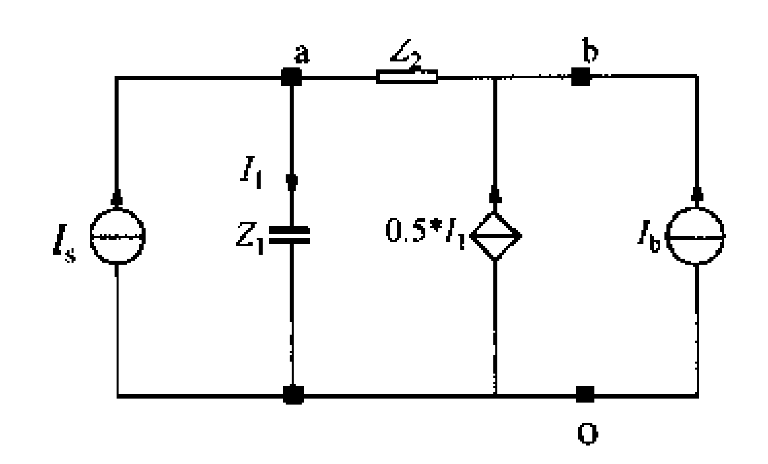


图 5.10-2　例 5.10 的原电路

解：■ 建模

本例可用戴维南定理求解，为此断开 b 端并接入外加电流源 $\dot I_b$，如图 5.10-2 所示。

列出节点方程和受控源的控制量与未知量 $\dot U_a$ 的关系为

$$\left(\frac{1}{Z_1}+\frac{1}{Z_2}\right)\dot U_a-\frac{1}{Z_2}\dot U_b=\dot I_s$$

$$-\frac{1}{Z_2}\dot U_a+\frac{1}{Z_2}\dot U_b=\dot I_b+0.5\dot I_1$$

$$\frac{1}{Z_1}\dot U_a=\dot I_1$$

整理以上方程，将未知量$\dot U_a$，$\dot U_b$， $\dot I_1$ 均移到等号左端，得

$$\begin{bmatrix} \frac{1}{Z_1}+\frac{1}{Z_2} & -\frac{1}{Z_2} & 0 \\ -\frac{1}{Z_2} & \frac{1}{Z_2} & -0.5 \\ \frac{1}{Z_1} & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{U}_a \\ \dot{U}_b \\ \dot{I}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{I}_s \\ \dot{I}_b \end{bmatrix}$$

令 $\dot{I}_b = 0$，$\dot{I}_s = 2\angle 0^\circ \text{A}$，得开路电压 $\dot{U}_{oc} = \dot{U}_b$

令 $\dot{I}_s = 0$，$\dot{I}_b = 1\angle 0^\circ$

得等效内阻抗 $Z_{eq} = \frac{\dot{U}_b}{\dot{I}_b} = \frac{\dot{U}_b}{1}$

于是得负载获取最大功率时 $Z_L = Z_{ep}$

最大功率为　$P_{L\max} = \frac{|\dot{U}_{oc}|^2}{4R_L}$

也可采用下面更简便的方法。

按图 5.10-2，由 b 经 Z_2，Z_1 列方程得

$$\begin{aligned} \dot{U}_b &= Z_2(\dot{I}_b + 0.5\dot{I}_1) + Z_1\dot{I}_1 \\ &= Z_2\dot{I}_b + (Z_1 + 0.5Z_2)\dot{I}_1 \end{aligned} \tag{5.6}$$

$$\dot{I}_1 = \dot{I}_s + \dot{I}_b + 0.5\dot{I}_1$$

即

$$\dot{I}_1 = 2\dot{I}_s + 2\dot{I}_b \tag{5.7}$$

代入式（5.6）得

$$\begin{aligned} \dot{U}_b &= Z_2 I_b + (Z_1 + 0.5Z_2)(2\dot{I}_s + 2\dot{I}_b) \\ &= (2Z_1 + 2Z_2)\dot{I}_b + (2Z_1 + Z_2)\dot{I}_s \end{aligned}$$

令　$\dot{I}_b = 0$，$\dot{I}_s = 2\angle 0^\circ$

得开路电压　$\dot{U}_{oc} = \dot{U}_b = (2Z_1 + Z_2)\dot{I}_s$

令　$\dot{I}_s = 0$，$\dot{I}_b = 1\angle 0^\circ$

得等效内阻抗　$Z_{eq} = \frac{\dot{U}_b}{\dot{I}_b} = \frac{\dot{U}_b}{1}$

$$Z_L = Z_{eq}$$

于是

$$P_{L\max} = \frac{|U_{oc}|^2}{4R_L}$$

■ MATLAB程序q510.m

```
clear, format compact
Z1=-j*250; Z2=250; ki=0.5; Is=2; % 设定元件参数
a11=1/Z1+1/Z2; a12=-1/Z2; a13=0; % 设定系数矩阵 A
a21=-1/Z2; a22=1/Z2; a23=-ki;
a31=1/Z1; a32=0; a33=-1;
A=[a11, a12, a13; a21, a22, a23; a31, a32, a33];
```

```
B=[1, 0; 0, 1; 0, 0];    % 设定系数矩阵 B
% 求方程解 X=[Ua; Ub; I1]=A\B*[Is; Ib];
X0=A\B*[Is; 0];
% Uoc 等于 Ib=0, Is=2∠0° 时的 Ub
Uoc=X0(2),
% Zeq 等于 Is=0, Ib=1 时的 Ub
X1=A\B*[0; 1];   Zeq=X1(2),
% 最大负载功率发生在 ZL=Zeq' 时
PLmax=(abs(Uoc))^2/4/real(Zeq)
```

■ **程序运行结果**

```
Uoc = 500 -1000i, abs(Uoc) = 1118V,
angle(Uoc) =-63.4349° ,
Zeq = 500-500iΩ
```

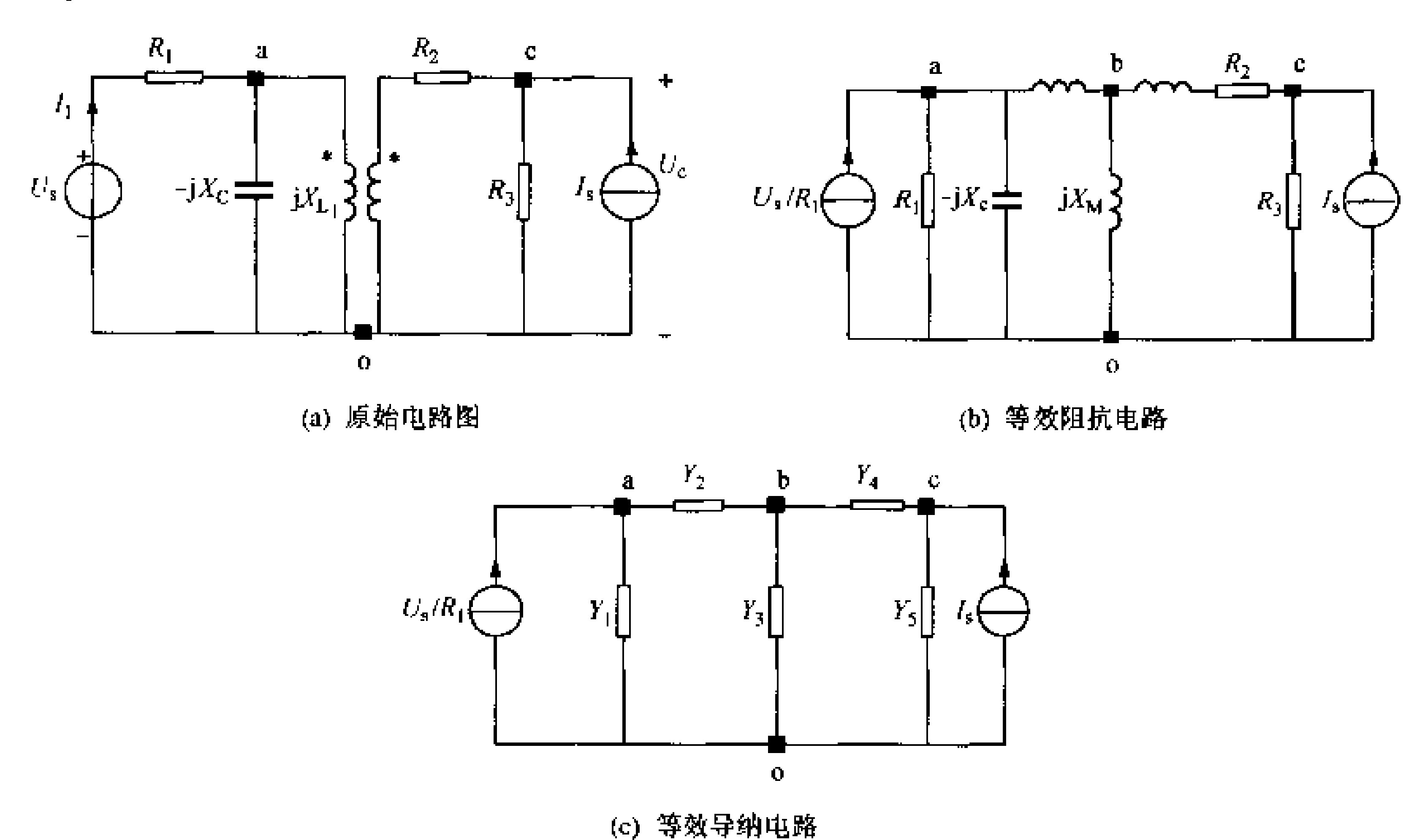


图 5.11　例 5.11 的等效电路变换

最大负载功率产生于 $Z_{Lpmax}=Z_{eq}'=500+500\mathrm{i}\,\Omega$ 时，此时 $P_{Lmax}=625\mathrm{W}$。

【例 5.11】 含互感的电路：复功率

如图 5.11(a)的电路，已知 $R_1=4\Omega$，

$R_2=R_3=2\Omega$， $X_{L_1}=10\Omega$， $X_{L_2}=8\Omega$，

$X_M=4\Omega$，$X_C=8\Omega$，

$\dot{U}_s=10\angle 0^\circ\mathrm{V}$，$\dot{I}_s=10\angle 0^\circ\mathrm{A}$。

求电压源、电流源发出的复功率。

解：■ 建模

为求得电源发出功率，需求得图 5.11（a）中的电流 I_1 和电流源端电压 U_c。

如利用节点法求解，可将互感电路变换为其去耦等效电路，同时将电压源变换为电流

源，如图 5.11(b)所示。按图 5.11(b)的简化电路图 5.11(c)可列出节点方程为

$$\begin{bmatrix} Y_1+Y_2 & -Y_2 & 0 \\ -Y_2 & Y_2+Y_3+Y_4 & -Y_4 \\ 0 & -Y_4 & Y_4+Y_5 \end{bmatrix}\begin{bmatrix} \dot{U}_a \\ \dot{U}_b \\ \dot{U}_c \end{bmatrix}=\begin{bmatrix} \dfrac{\dot{U}_s}{R_1} \\ 0 \\ \dot{I}_s \end{bmatrix} \tag{5.8}$$

式中

$$Y_1=\frac{1}{R_1}+\frac{1}{(-\mathrm{j}X_c)}, \quad Y_2=\frac{1}{\mathrm{j}(X_{L_1}-X_M)}$$

$$Y_3=\frac{1}{\mathrm{j}X_M}, \quad Y_4=\frac{1}{R_2+\mathrm{j}(X_{L_2}-X_M)}$$

$$Y_5=\frac{1}{R_3}$$

由式（5.8）解得　$\dot{U}_a$和$\dot{U}_c$。

由图5.11(a) 可见　$\dot{I}_1=\dfrac{\dot{U}_s-\dot{U}_a}{R_1}$。

于是可得，电压源发出复功率 $S_{U_s}=\dot{U}_s\dot{I}^*$，电流源发出复功率 $S_{I_s}=\dot{U}_c\dot{I}_s^*$。

■ MATLAB程序q511.m

```
clear, format compact
R1=4; R2=2; R3=2; % 设定元件参数
Xl1=10; Xl2=8; XM=4; Xc=8;
Us=10; Is=10; % 设定信号源参数
Y1=1/R1+1/(-j*Xc); Y2=1/(j*(Xl1-XM)); Y3=1/(j*XM);
Y4=1/(R2+j*(Xl2-XM)); Y5=1/R3;
a11=Y1+Y2; a12=-Y2; a13=0;          % 设定系数矩阵 A
a21=-Y2; a22=Y2+Y3+Y4; a23=-Y4;
a31=0; a32=-Y4; a33=Y4+Y5;
A=[a11, a12, a13; a21, a22, a23; a31, a32, a33];
B=[Us/R1; 0; Is];                   % 设定系数矩阵 B
X=A\B;
% X=[Ua; Ub; Uc]=A\[Us/R1; 0; Is];
I1=(Us-X(1))/R1;
Pus=Us*I1', Pis=X(3)*Is'
% 求 Us 和 Is 产生的复功率
```

■ 程序运行结果

```
X = [Ua;  Ub;  Uc]
  11.6195 - 3.7532i
   8.5347 + 1.4910i
  17.5064 + 3.2391i
```

```
Pus = -4.0488 - 9.3830i
Pis = 1.7506e+002 +3.2391e+001i
```

【例 5.12】 正弦稳态电路：求未知参数

如图 5.12 的电路，已知 U_s=100V，I_1=100mA， 电路吸收功率 P=6W，X_{L_1}=1250Ω，X_{c_2}=750Ω，电路呈感性，求 R_3 及 X_{L_3}。

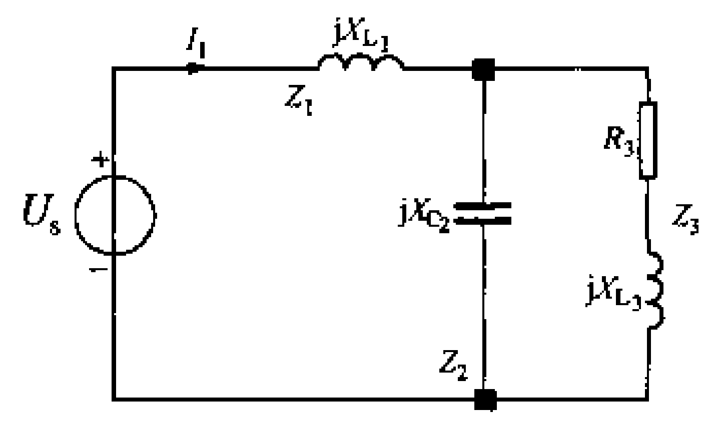


图 5.12 例 5.12 电路图

解：■ 建模

设电源端的总阻抗

$$Z = R + jX$$

由图5.12知 $Z = Z_1 + \dfrac{1}{Y_2 + Y_3}$，$Y_2 = \dfrac{1}{Z_2}$，$Y_3 = \dfrac{1}{Z_3}$

总阻抗的模 $|Z| = \dfrac{U_s}{I}$，

由于 Z_1，Z_2 为纯电抗元件，它们不吸收有功功率，故 $P=I^2R$（R 为总电阻），即

$$R = \frac{P}{I^2}$$

因

$$|Z| = \sqrt{R^2 + X^2}$$

$$X = \pm\sqrt{|Z|^2 - R^2}$$

得知电路呈感性，取"+"号，即

$$Z = R + jX$$

Z_2，Z_3 的并联阻抗 $Z_{23}=Z-Z_1=Z-jX_{L_1}$

其倒数 $\dfrac{1}{Z_{23}} = Y_2 + Y_3$

$$Y_3 = \frac{1}{Z_{23}} - Y_2 = \frac{1}{Z_{23}} - \frac{1}{-jX_c}, \quad Z_3 = \frac{1}{Y_3}$$

最后得

$$R_3 = \mathrm{Re}[Z_3]$$

$$X_{L3} = \mathrm{Im}[Z_3]$$

■ MATLAB程序q512.m

```
clear, format compact
XL1=1250; Xc2=750;                    % 设定元件参数
Us=100; I=100; P=6;                   % 设定信号源参数
Z1=j*XL1; Z2=-j*Xc2;                  % 求两已知支路的阻抗
R=P/I^2; absZ=Us/I;                   % 求电路的总阻抗的模
absX=sqrt(absZ^2-R^2);                % 求电路的总电抗
Z=R+j*absX;                           % 求电路的总阻抗
Z23=Z-Z1;                             % 求 2, 3 支路并联的阻抗
Y3=1/Z23-1/Z2; Z3=1/Y3                % 求 3 支路的阻抗
R3=real(Z3), XL3=imag(Z3)             % 求 3 支路的电阻和电抗
```

■ 程序运行结果

```
Z3 = 750    375i
R3 = 750Ω, XL3 = 375Ω
```

【例 5.13】 正弦稳态电路：利用模值求解

图 5.13 所示电路中，已知 I_R=10A，X_c=10Ω，并且 U_1=U_2=200V，求 X_L。

解：■ 建模

本例中 U_1，U_2，只知有效值，初相未知，故用模值求解较为方便。列出 U_2 的节点方程为：

$$(Y_1+Y_2+Y_3)U_2=Y_1U_1$$

式中
$$Y_1=\mathrm{j}\frac{1}{X_c},\ Y_2=-\mathrm{j}\frac{1}{X_1},\ Y_3=\frac{1}{R}=\frac{I_r}{U_2}=\frac{1}{20}$$

等号两端同除以 U_2 并取模得

$$|Y_1+Y_2+Y_3|=\left|Y_1\frac{\dot{U}_1}{\dot{U}_2}\right|$$

由于
$$Y_1+Y_2+Y_3=\frac{1}{R}-\mathrm{j}\left(\frac{1}{X_1}-\frac{1}{X_c}\right)$$

$$|Y_1+Y_2+Y_3|=\sqrt{\left(\frac{1}{R}\right)^2+\left(\frac{1}{X_1}-\frac{1}{X_c}\right)^2}=\left|Y_1\frac{\dot{U}_1}{\dot{U}_2}\right|=\frac{1}{X_c}$$

可解得：
$$Y_1=\frac{1}{X_1}=\frac{1}{X_c}\pm\sqrt{\left(\frac{1}{X_c}\right)^2-\left(\frac{1}{R}\right)^2}$$

$$X_1=\frac{1}{Y_1}$$

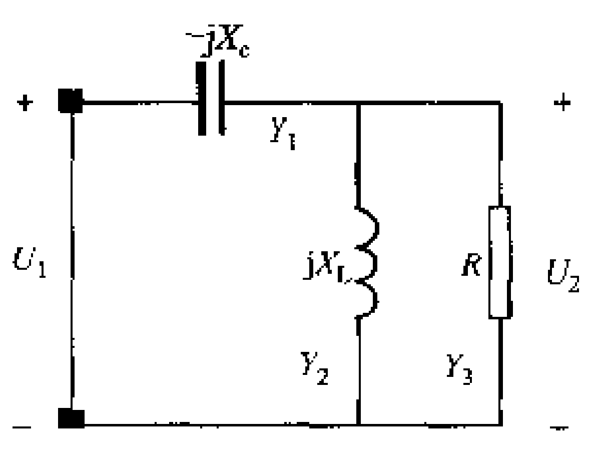


图 5.13　例 5.13 的电路图

■ MATLAB程序q513.m

```
IR=10; Xc=10;                % 设定元件值
U1=200; U2=200;              % 设定已知变量
R=U2/IR              %求 R
YL(1)=1/Xc+sqrt((1/Xc)^2-(1/R)^2);
YL(2)=1/Xc-sqrt((1/Xc)^2-(1/R)^2)
XL=1./YL
```

■ 程序运行结果

```
YL = 0.1866      0.0134
XL = 5.3590      74.6410
```

5.4　频率响应

频率响应函数（即正弦稳态网络函数）定义为响应（输出）相量 $\dot{Y}$ 与激励（输入）相量 $\dot{F}$ 之比，即 $H(\mathrm{j}\omega)=|H(\mathrm{j}\omega)|\mathrm{e}^{\mathrm{j}\theta(\omega)}=\dfrac{\dot{Y}}{\dot{F}}$。

用 MATLAB 中的 abs(H)和 angle(H)语句可直接计算幅频响应和相频响应，而且其图形的频率坐标（横坐标）可以是线性的（用 plot），也可以是半对数的(用 semilogx)，这给计算

和绘制幅、相特性带来很大方便。

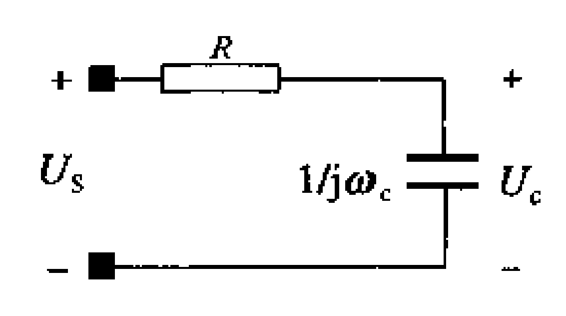


图 5.14-1　例 5.14 的电路图

【例 5.14】 一阶低通电路的频率响应

图 5.14-1 是一阶 RC 低通电路，若以 $\dot{U}_c$ 为响应，求频率响应函数，画出其幅频响应（幅频特性）$|H(j\omega)|$ 和相频的响应（相频特性）$\theta(\omega)$

解：■ 建模

由图 5.14-1 用分压公式不难求得频率响应函数

$$H(j\omega)=\frac{\dot{U}_c}{\dot{U}_s}=\frac{-j\dfrac{1}{\omega C}}{R-j\dfrac{1}{\omega C}}=\frac{1}{1+j\omega CR}=\frac{1}{1+j\dfrac{\omega}{\omega_c}}$$

式中 $\omega_c=\dfrac{1}{RC}$ 为截止角频率。

设无量纲频率 $\omega_w=\dfrac{\omega}{\omega_c}$ =0，0.2，0.4，…，4，画出幅频响应及相频的响应。

■ MATLAB程序q514.m

```
clear, format compact
ww=0:0.2:4;                          % 设定频率数组 ww=w/wc
H=1./(1+j*ww);                                  % 求复频率响应
figure(1)
subplot(2,1,1), plot(ww,abs(H)),   % 绘制幅频特性
grid, xlabel('ww'), ylabel('abs(H)')
subplot(2,1,2), plot(ww,angle(H))       % 绘制相频特性
grid, xlabel('ww'), ylabel('angle(H)')
figure(2)                    % 绘制对数频率特性
subplot(2,1,1), semilogx(ww,20*log10(abs(H))) %纵坐标为分贝
grid, xlabel('ww'), ylabel('分贝')
subplot(2,1,2), plot(ww,angle(H))% 绘制相频特性
grid, xlabel('ww'),
ylabel('angle(H)')
```

■ 程序运行结果

如图 5.14-2 所示。

【例 5.15】 频率响应：二阶低通电路

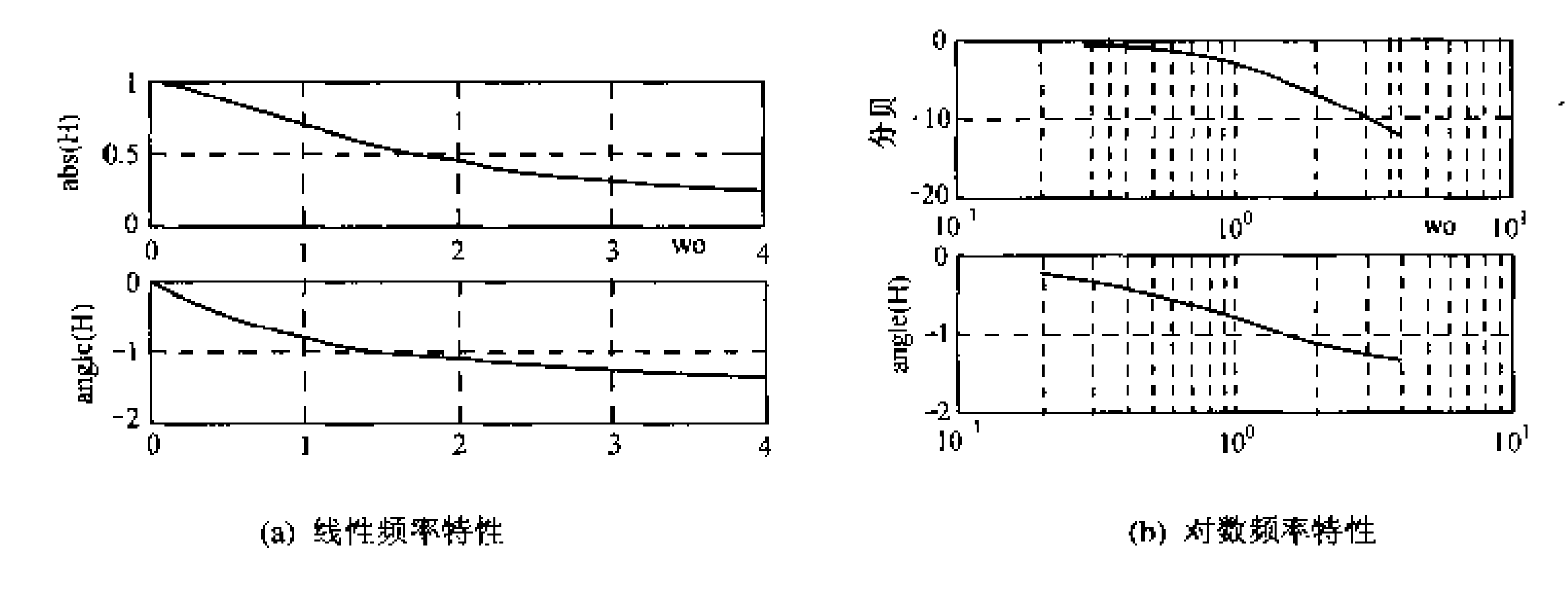
二阶低通函数的典型形式为

$$H(j\omega)=\frac{\dot{U}_2}{\dot{U}_1}=H_0\frac{\omega_n^2}{s^2+\dfrac{\omega_n}{Q}s+\omega_n^2} \tag{5.9}$$

式中 $s=j\omega$。

令 $H_0=1$，画出 $Q=\dfrac{1}{3}$，$\dfrac{1}{2}$，$\dfrac{1}{\sqrt{2}}$，1，2，5 的幅频相频响应。当 $Q=\dfrac{1}{\sqrt{2}}$ 时，称为最平幅

度特性（即 Butterworth 特性），即在通带内其幅频特性最平坦。



(a) 线性频率特性　　(b) 对数频率特性

图 5.14-2　例 5.14 的频率特性

解：■ 建模

令 H_0=1，s=jω，式（5.9）可简化为

$$H(\mathrm{j}\omega)=\frac{1}{1-\left(\dfrac{\omega}{\omega_n}\right)^2+\mathrm{j}\dfrac{1}{Q}\dfrac{\omega}{\omega_n}}$$

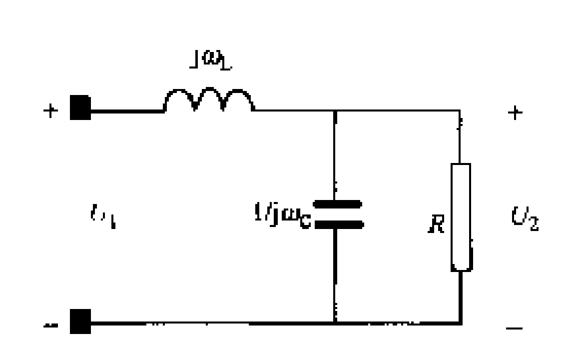


图 5.15-1　例 5.15 的电路图

幅频响应若用增益表示为　$G=20\log|H(\mathrm{j}\omega)|$

相频特性　$\theta(\omega)=\angle H(\mathrm{j}\omega)$

横坐标用对数无量纲频率，取

$\omega_w=\dfrac{\omega}{\omega_n}=0.1$，…，1，…，10

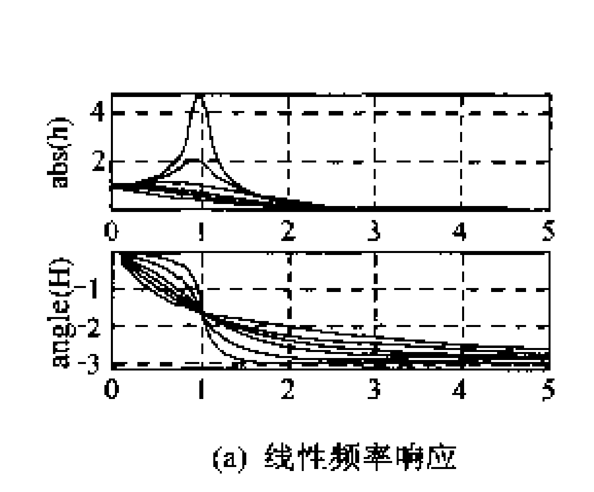
令 $Q=\dfrac{1}{3}$，$\dfrac{1}{2}$，$\dfrac{1}{\sqrt{2}}$，1，2，5 画图。

■ MATLAB程序q515.m

```
clear, format compact
for Q=[1/3, 1/2, 1/sqrt(2), 1, 2, 5]
   ww=logspace(-1, 1, 50);              % 设定无量纲频率数组 ww=w/w0
   H=1./(1+j*ww/Q+(j*ww).^2);     % 求复频率响应
   figure(1)
   subplot(2, 1, 1), plot(ww, abs(H)), hold on          % 绘制幅频特性
   subplot(2, 1, 2), plot(ww, angle(H)), hold on        % 绘制相频特性
   figure(2)                    % 绘制对数频率特性
   subplot(2, 1, 1), semilogx(ww, 20*log10(abs(H))), hold on %纵坐标为分贝
   subplot(2, 1, 2), semilogx(ww, angle(H)), hold on     % 绘制相频特性
end
figure(1),subplot(2, 1, 1),     grid, xlabel('w'), ylabel('abs(H) ')
subplot(2, 1, 2),     grid, xlabel('w'), ylabel('angle(H) ')
figure(2),subplot(2, 1, 1),     grid, xlabel('w'), ylabel('分贝')
subplot(2, 1, 2),     grid, xlabel('w'), ylabel('angle(H) ')
```

■ **程序运行结果**

如图 5.15-2 所示。



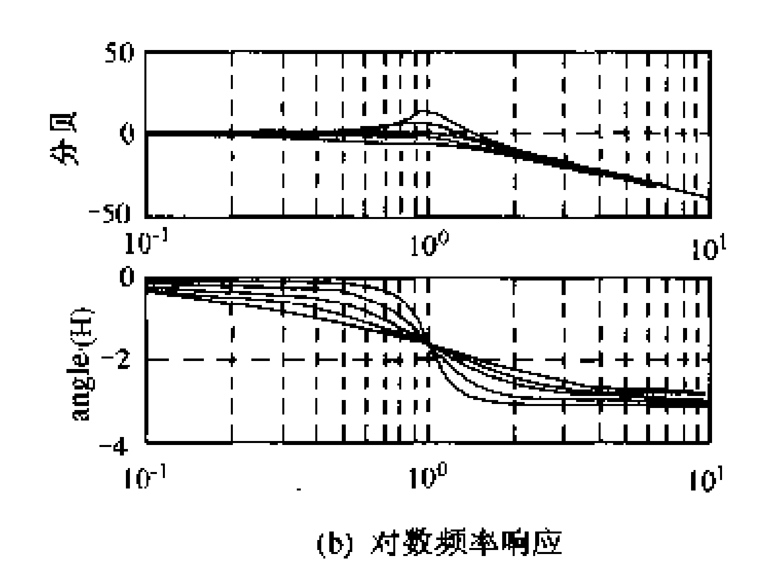


图 5.15-2　频率响应

【例 5.16】 频率响应：二阶带通电路

图 5.16-1 是互相对偶的串联和并联谐振电路，其频率响应函数为

$$H(\mathrm{j}\omega)=\frac{H_0}{1+\mathrm{j}Q\left(\dfrac{\omega}{\omega_n}-\dfrac{\omega_n}{\omega}\right)}$$

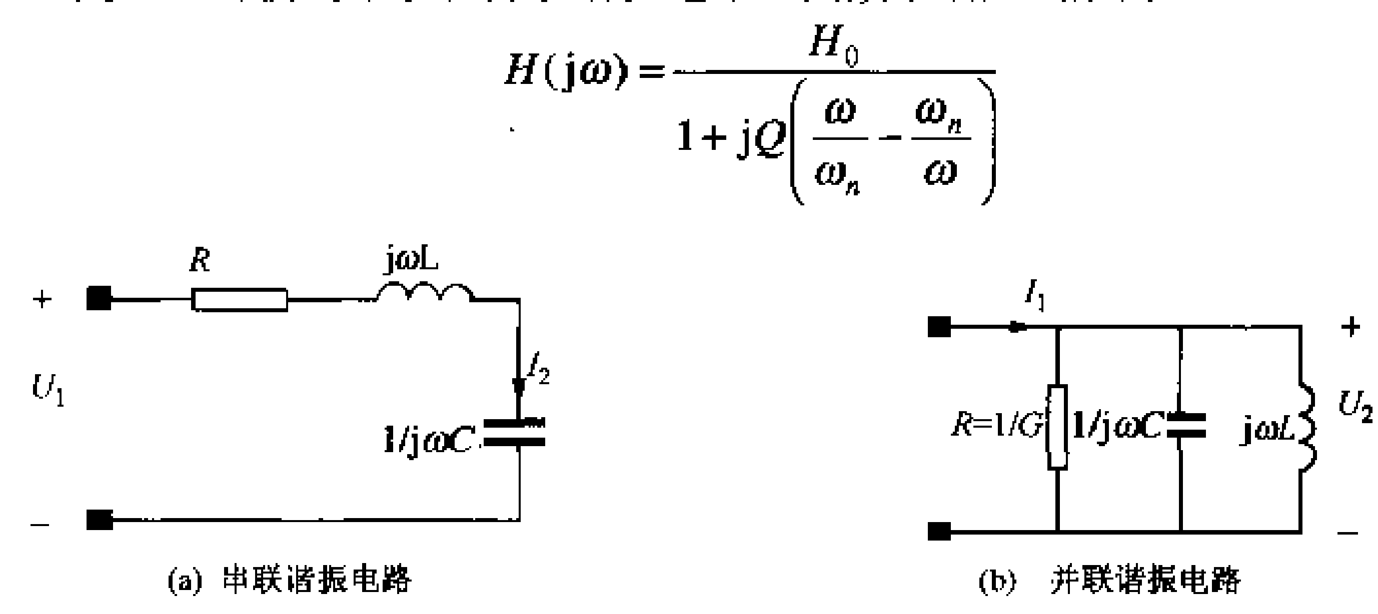


图 5.16-1　谐振电路

各电路的参数分别为

(a) 串联谐振电路

$$H(\mathrm{j}\omega)=\frac{\dot{I}_2}{\dot{U}_1}$$

$$\omega_n^2=\frac{1}{LC}$$

$$Q=\frac{\omega_n L}{R}$$

$$H_0=\frac{1}{R}$$

(b) 并联谐振电路

$$H(\mathrm{j}\omega)=\frac{U_2}{I_1}$$

$$\omega_n^2=\frac{1}{LC}$$

$$Q=\frac{\omega_n C}{G}=\omega_n CR$$

$$H_0=\frac{1}{G}=R$$

令 H_0=1，画出 Q=5，10，20，50，100 的幅频和相频响应。

解：■ 建模

幅频响应若用增益表示为　　$G(\omega)=20\log|H(\mathrm{j}\omega)|$

相频响应　　$\theta(\omega)=\mathrm{angle}(H(\mathrm{j}\omega))$

横坐标用对数坐标取

$$\frac{\omega}{\omega_n}=0.1,\cdots,1,\cdots,10$$

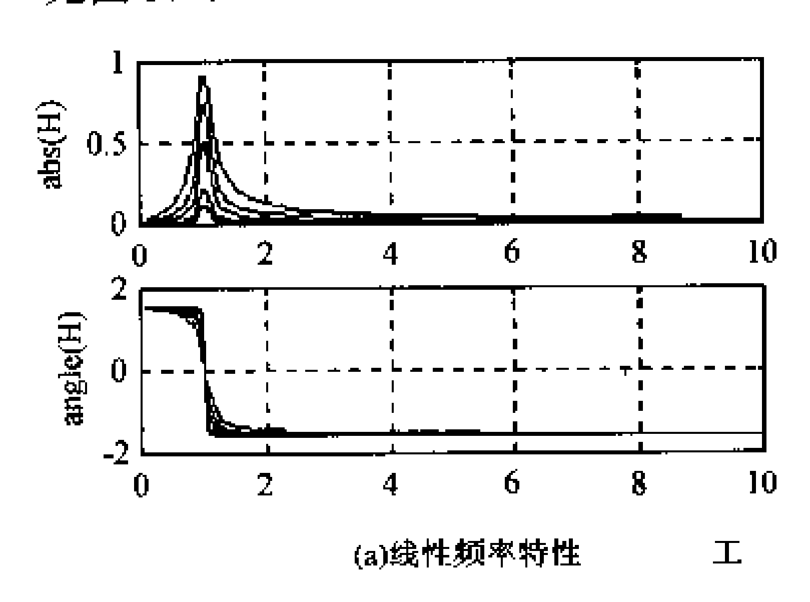
取　Q=5，10，20，50，100，做图

■ MATLAB 程序 q516.m

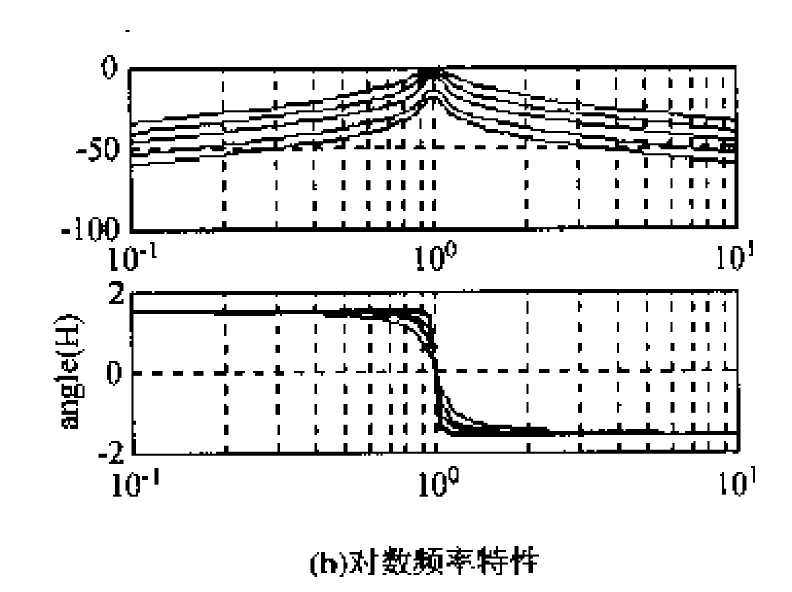
```
H0=1; Wn=1;
for Q=[5, 10, 20, 50, 100]
   w=logspace(-1, 1, 50);                  % 设定频率数组 w
   H=H0./(1+j*Q*(w/w0-Wn./w));             % 求复频率响应
   figure(1)              %横坐标为线性坐标绘制频率特性
   subplot(2, 1, 1), plot(w, abs(H)), hold on       % 绘制幅频特性
   subplot(2, 1, 2), plot(w, angle(H)), hold on% 绘制相频特性
   figure(2)                               % 绘制对数频率特性
   subplot(2, 1, 1), semilogx(w, 20*log10(abs(H))), hold on %纵坐标为分贝
   subplot(2, 1, 2), semilogx(w, angle(H)), hold on      % 绘制相频特性
end
```

■ 程序运行结果

见图 5.16-2。



(a)线性频率特性



(b)对数频率特性

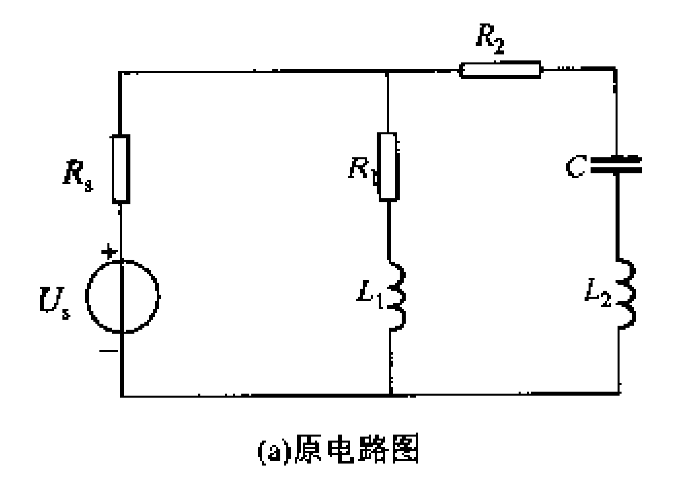
图 5.16-2　谐振回路的频率特性

【例 5.17】 复杂谐振电路的计算

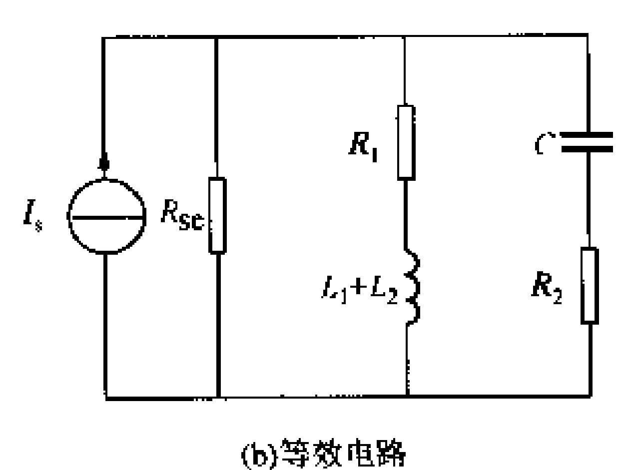
图 5.17-1 为一双电感并联电路，已知：R_s=28.2 kΩ，R_1=2Ω，R_2=3Ω，L_1=0.75mH，L_2=0.25mH，C=1000pF。求回路的通频带 B 及满足回路阻抗大于 50 kΩ 的频率范围。

解：■ 建模

先把回路变换为一个等效单电感谐振回路，把信号源的内阻 R_s 变为并接在该单电感回路上的等效内阻 R_{se}，如图 5.17-1(b)所示。按照这个等效电路写出的方程如下。



(a)原电路图



(b)等效电路

图 5.17-1　例 5.17 的等效电路变换

设　$m=\dfrac{L_1}{L_1+L_2}$，则

$$R_{se}=\frac{R_s}{m^2},\quad I_s=m\frac{U_s}{R_s}$$

其他两支路的等效阻抗分别为

$$Z_{1e}=R_1+\mathrm{j}\omega(L_1+L_2),\ \ Z_{2e}=R_2+\frac{1}{\mathrm{j}\omega C}$$

总阻抗是三个支路阻抗的并联

$$Z_e=\left(\frac{1}{R_{se}}+\frac{1}{Z_{1e}}+\frac{1}{Z_{2e}}\right)^{-1}$$

其谐振曲线可按 Z_e 的绝对值直接求得。

■ MATLAB 程序 q517.m

```
R1=2;  R2=3;  L1=0.75e-3;  L2=0.25e-3;  C=1000e-12;  Rs=28200;
L=L1+L2; R=R1+R2; Rse=Rs*(L/L1)^2;          % 折算内阻
f0=1/(2*pi*sqrt(C*L))
Q0=sqrt(L/C)/R, R0=L/C/R;                    % 空载 Q0 值
Re=R0*Rse/(R0  Rse),                         % 折算内阻与回路电阻的并联
Q=Q0*Re/R0, B=f0/Q,                          % 实际 Q 值和通带
s=log10(f0);
f=logspace(s-.1, s+.1, 501); w=2*pi*f;          % 设定计算的频率范围及数组
z1e=R1+j*w*L;  z2e=r2+1./(j*w*C);      % 等效单回路中两个电抗支路的阻抗
ze =1./(1./z1e+1./z2e+1./Rse);               % 等效单回路中三个支路的并联阻抗
subplot(2, 1, 1), loglog(w, abs(ze)), gRid % 画对数幅频特性
axis([min(w), max(w), 0.9*min(abs(ze)), 1.1*max(abs(ze))])
subplot(2, 1, 2), semilogx(w, angle(ze)*180/pi)          % 画相频特性
axis([min(w), max(w), -100, 100]), grid
fh=w(find(abs(1./(1./z1e  1./z2e))>50000))/2/pi;   % 求幅频特性大于 50kHz 的频带
fhmin=min(fh), fhmax=max(fh),
```

■ 程序运行结果

谐振频率 f0 = 159.15 kHz

空载品质因数 Q0 = 200

等效信号源内阻 Rse = 5.0133e+004

考虑内阻后的品质因数 Q = 40.0853

通频带 B = 3.9704e+003

回路阻抗大于 50 kΩ 的频率范围

fhmin = 157.7kHz

fhmax = 160.63kHz

谐振频率附近的幅频和相频特性曲线见图 5.17-2。

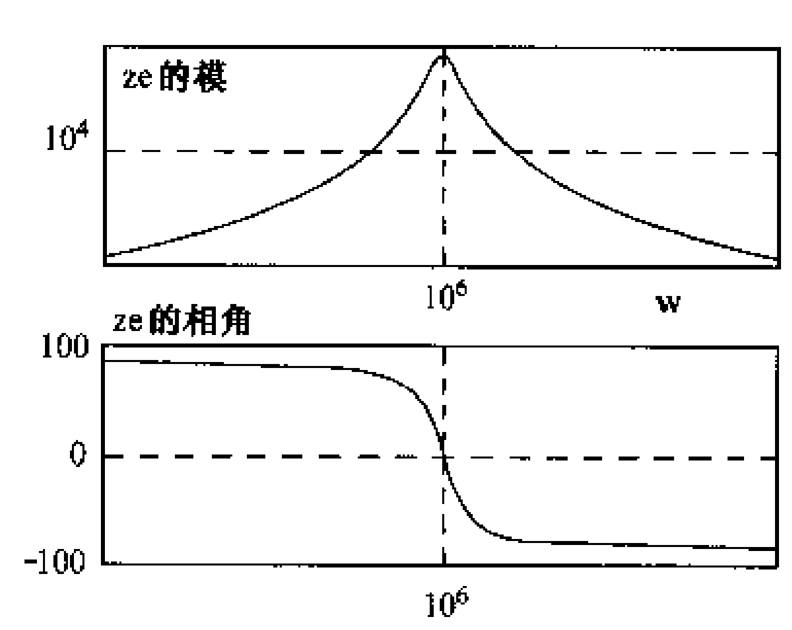


图 5.17-2　谐振频率处的幅频和相频特性

5.5　二端口电路

二端口电路参数（见图 5.18-1）的互相转换和网络函数的计算较为繁杂，且易出错，特别是当参数为复数时，更是如此。而 MATLAB 的复数矩阵运算非常方便。可将公式列写出来，以便编程时调用。

5.5.1　Z，Y，H，G，A，B 六种参数间关系的 MATLAB 语句

● Z 与 Y，H 与 G，A 与 B 的关系。

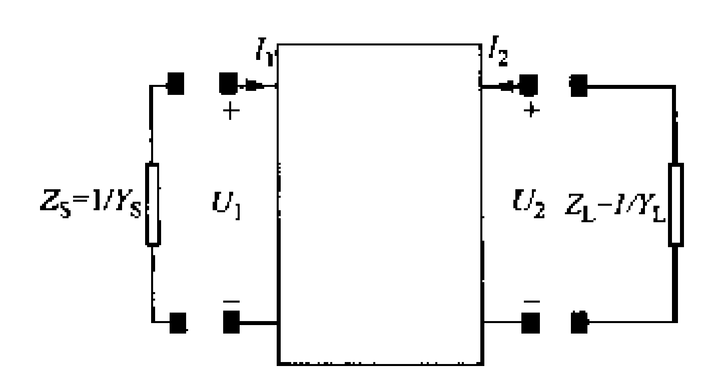


图 5.18-1　二端口电路的参数

- Z=inv (Y)，　Y=inv (Z)
- H=inv (G)，　G=inv (H)
- A=[B(2，2)，B(1，2)；B(2，1)，B(1，1)]/det(B)
- B=[A(2，2)，A(1，2)；A(2，1)，A(1，1)]/det(A)

● 由 Z 求 A 或 H

- A=[Z(1，1)，det(z)；1，Z(2，2)]/Z(2，1)
- H=[det(Z)，Z(1，2)；−Z(2，1)，1]/Z(2，2)

● 由 A 求 Z 或 H

- Z=[A(1，1)，det(A)；1，A(2，2)]/A(2，1)
- H=[A(1，2)，det(A)；−1，A(2，1)]/A(2，2)

● 由 H 求 Z 或 A

- Z=[det(H)，H(1，2)；−H(2，1)，1]/H(2，2)
- A=−[det(H)，H(1，1)；H(2，2)，1]/H(2，1)

有了这些关系，则任意一种参数都可相互转换。

5.5.2　网络函数及其 MATLAB 语句

以下各式中，$Z_L\left(=\frac{1}{Y_L}\right)$为负载阻抗，$Z_s\left(=\frac{1}{Y_s}\right)$为输入端接阻抗，$\Delta_z=z_{11}z_{22}-z_{12}z_{21}$

● 输入阻抗，负载端接 Z_L，即有$U_2=-Z_LI_2$

$$Z_{in}=\frac{U_1}{I_1}=\frac{\Delta_z+z_{11}Z_L}{z_{22}+Z_L}=\frac{a_{11}Z_L+a_{12}}{a_{21}Z_L+a_{22}}$$

● 输出阻抗，输入端接 Z_s，即有$U_1=-Z_sI_1$

$$Z_{out}=\frac{U_2}{I_1}=\frac{\Delta_z+z_{22}Z_s}{z_{11}+Z_s}=\frac{a_{22}Z_s+a_{12}}{a_{21}Z_s+a_{11}}$$

● 电压比（负载端接 Z_L）

$$A_u=\frac{U_2}{U_1}=\frac{z_{21}Z_L}{\Delta_z+z_{11}Z_L}=\frac{Z_L}{a_{11}Z_L+a_{12}}$$

● 电流比（负载端接 Z_L）

$$A_i=\frac{I_2}{I_1}=\frac{-z_{21}}{z_{22}+Z_L}=\frac{-1}{a_{21}Z_L+a_{22}}$$

● 转移阻抗（负载端接 Z_L）

$$Z_T=\frac{U_2}{I_1}=\frac{z_{21}Z_L}{z_{22}+Z_L}=\frac{Z_L}{a_{21}Z_L+a_{22}}$$

● 转移导纳（负载端接 Z_L）

$$Y_T=\frac{I_2}{U_1}=\frac{-z_{21}}{\Delta_z+z_{11}Z_L}=\frac{-1}{a_{11}Z_L+a_{12}}$$

■ **供拷贝的 MATLAB 程序 q518f.m**

```
% 以下的公式不是为了直接执行，而是为读者复制和粘贴之用
% 以下各式中，ZL=1/YL 为负载阻抗，ZS=1/YS 为输入端信号源阻抗
% (1)输入阻抗（负载端接 ZL，即有 U2=-ZL*I2）
%    Zin=U1/I1=(A(1, 1)*ZL+A(1, 2))/(A(2, 1)*ZL+A(2, 2));
% (2)输出阻抗（输入端接 ZS，即有 U1=-ZS*I1）
%    Zout=U2/I2=(A(2, 2)*ZS+A(1, 2))/(A(2, 1)*ZS+A(1, 1));
%    电压比（负载端接 ZL)
% Au=U2/U1=Z(2, 1)*ZL/(det(Z)+Z(1, 1)*ZL)=ZL/(A(1, 1)*ZL+A(1, 2));
%    电流比（负载端接 ZL)
% Ai=I2/I1=-Z(2, 1)/(Z(2, 2)+ZL)=-1/(A(2, 1)*ZL+A(2, 2));
%    转移阻抗（负载端接 ZL)
% ZT=U2/I1=Z(2, 1)*ZL/(Z(2, 2)+ZL)=ZL/(A(2, 1)*ZL+A(2, 2));
%    转移导纳（负载端接 ZL)
% YT=I2/U1=-Z(2, 1)/(det(Z)+Z(1, 1)*ZL)=-1/(A(1, 1)*ZL+A(1, 2));
```

【例 5.18】 网络参数的计算与变换

图 5.18-2 的二端口网络，R=100Ω；L=0.02H；C=0.01F，频率ω=300rad/s，求其 Y 参数及 H 参数。

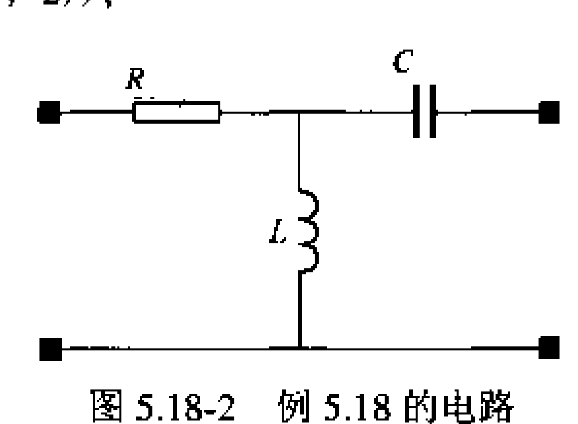


图 5.18-2 例 5.18 的电路

解：■ **MATLAB 程序 q518.m**

```
format long
R=100;  L=0.02;  C=0.01; w=300;
z1=R;  z2 =j*w*L;  z3=1/(j*w*C);
Z(1, 1)=z1+z2; Z(1, 2)=z2;  Z(2, 1)=z2;  Z(2, 2)=z2+z3;
Y=inv(Z),
H=[det(Z),  Z(1, 2);  -Z(2, 1), 1] / Z(2, 2)
```

■ **程序运行结果**

```
Y =
   0.0099998754      0.0000352937i     -0.0105881034  - 0.0000373698i
  -0.0105881034     -0.0000373698i      0.0112109330  - 0.1764310202i
H =
   100.00000000     -0.3529411765i      1.0588235294    0i
    -1.0588235294    0i                 0              - 0.1764705882i
```

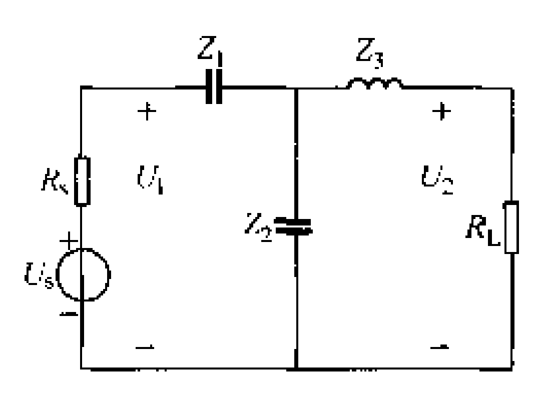


图 5.19　例 5.19 的电路图

【例 5.19】 阻抗匹配网络的计算

为使信号源（其内阻 R_s=12Ω）与负载（R_L=3Ω）相匹配，在其间插入一阻抗匹配网络，如图 5.19 所示，已知 Z_1=−j6Ω，Z_2=−j10Ω，Z_3=j6Ω。若 $U_s=24\angle0^\circ$，求负载吸收的功率。

解：■ 建模

方法 1　用 Z 方程求解，对于二端口电路有

$$\dot{U}_1=z_{11}\dot{I}_1+z_{12}\dot{I}_2 \quad \text{即} \quad \dot{U}_1-z_{11}\dot{I}_1-z_{12}\dot{I}_2=0$$

$$\dot{U}_2=z_{21}\dot{I}_1+z_{22}\dot{I}_2 \quad \text{即} \quad \dot{U}_2-z_{21}\dot{I}_1-z_{22}\dot{I}_2=0$$

对电源端有　$\dot{U}_s=R_s\dot{I}_1+\dot{U}_1$　即　$\dot{U}_1+R_s\dot{I}_1=\dot{U}_s$

对负载端有　$\dot{U}_2=-R_2\dot{I}_2$　即　$\dot{U}_2+R_2\dot{I}_2=0$

将以上四式写为矩阵形式为

$$\begin{bmatrix}1 & 0 & -z_{11} & -z_{12}\\0 & 1 & -z_{21} & -z_{22}\\1 & 0 & R_s & 0\\0 & 1 & 0 & R_L\end{bmatrix}\begin{bmatrix}\dot{U}_1\\\dot{U}_2\\\dot{I}_1\\\dot{I}_2\end{bmatrix}=\begin{bmatrix}0\\0\\\dot{U}_s\\0\end{bmatrix} \tag{5.10}$$

其中　$z_{11}=Z_1+Z_2=-\text{j}16$，$z_{12}=-\text{j}10$

$z_{22}=Z_2+Z_3=-\text{j}4$，$z_{21}=-\text{j}10$

$R_s=12\Omega$，$R_L=3\Omega$，$\dot{U}_s=24\angle0^\circ\text{V}$

解出 U_2，则负载吸收功率为　$P=\dfrac{|\dot{U}_2|^2}{R_L}$

方法 2　应用戴维南定理求解。方程(1) 中，令 $\dot{I}_2=0$，可得开路电压 $\dot{U}_{oc}=\dot{U}_2|_{\dot{I}_2=0}$，当 $\dot{U}_s=0$ 时，负载端的输出阻抗即为等效内阻抗

$$Z_{eq}=Z_{out}=\frac{\Delta_z+z_{22}R_s}{z_{11}+R_s}$$

按戴维南等效电路，得负载吸收功率

$$P=\left|\frac{\dot{U}_{oc}}{Z_{out}+R_L}\right|^2R_L$$

■ MATLAB 程序 q519.m

```
clear, format         % 调用画线路图函数
Rs=12; RL=3;
Z1=-6j; Z2=-10j; Z3=6j; Us=24;
% 方法 1 用 Z 方程求解
Z(1, 1)=Z1+Z2; Z(1, 2)=-10j;        % 列出 Z 矩阵各分量
Z(2, 1)=Z(1, 2); Z(2, 2)=Z2+Z3;
% 系数矩阵 A 和系数矩阵 B
A=[1, 0,-Z(1,1),-Z(1,2); 0, 1,-Z(0,1),-Z(2,2); 1, 0, Rs, 0; 0, 1, 0, RL];
B=[0; 0; Us; 0];     X=A\B                          % 求方程解
```

```
U1=X(1); U2=X(2); I1=X(3); I2=X(4);    % 列出各未知数的解
P=abs(U2)^2/RL                          % 求负载功率
%方法 2、用戴维南定理求解
Uoc=Us*Z2/(Z2+Rs+Z1)                    % 等效电压源开路电压
Zout=(det(Z)+Z(2, 2)*Rs)/(Z(1, 1)+Rs)      % 等效电压源内阻
P1=abs(Uoc/(Zout+RL))^2*RL              % 求负载功率
```

■ 程序运行结果

● 方法 1

```
X =  [U1; U2; I1; I2] =12.0000
                       4.8000 - 3.6000i
                       1.0000 - 0.0000i
                      -1.6000 + 1.2000i
P =12.0000
```

● 方法 2

```
Uoc =   9.6000 - 7.2000i
Zout =  3
P1 =    12
```

【例 5.20】 桥梯形全通网络的计算

图 5.20-1 中的二端口网络是定阻全通网络（定阻指接以电阻性负载时，在所有频率下，输入阻抗为常数，全通指幅频响应为常数，其相频响应可按要求进行设计）。将它插入级联网络中只改变相频响应而不影响幅频响应。

已知 $C_1=C_3=2\text{F}$，$L_2=1\text{H}$，$C_2=4/3\text{F}$，$L_4=1\text{H}$，$R_L=1\,\Omega$。求其网络函数

$$H(\mathrm{j}\omega)=\frac{U_2}{U_1}$$

和输入阻抗 Z_{in}，并画出幅频相频响应及输入阻抗。

解：■ 建模

图 5.20-1 的桥 T 形网络可看做是两个子网络 N_a，N_b 相并联，如图 5.20-2 所示，设其 $\boldsymbol{Y}$ 参数分别为 $\boldsymbol{Y}_a$ 和 $\boldsymbol{Y}_b$，则桥 T 形网络的 $\boldsymbol{Y}$ 矩阵为两者之和。

对图 5.20-2 中的子电路 N_a 有

$$I_{1a}=\frac{U_1-U_2}{Z_4}=\frac{1}{Z_4}U_1-\frac{1}{Z_4}U_2$$

电路是互易且对称的，故得 N_a 的 $\boldsymbol{Y}$ 矩阵为

$$\boldsymbol{Y}_a=\begin{bmatrix}\dfrac{1}{Z_4} & \dfrac{-1}{Z_4}\\ \dfrac{-1}{Z_4} & \dfrac{1}{Z_4}\end{bmatrix}$$

对图 5.20-2 中的子电路 N_b，容易求得其 $\boldsymbol{Z}$ 矩阵为

$$\boldsymbol{Z}_b=\begin{bmatrix}Z_1+Z_2 & Z_2\\ Z_2 & Z_3+Z_2\end{bmatrix}$$

故　$\boldsymbol{Y}_b=\boldsymbol{Z}_b^{-1}$

其中 $$Z_1 = Z_3 = -\mathrm{j}\frac{1}{2\omega},\quad Z_2 = \mathrm{j}\left(\omega - \frac{3}{4\omega}\right)$$

$$Z_4 = \mathrm{j}\omega,\quad Z_L = R_L = 1\Omega$$

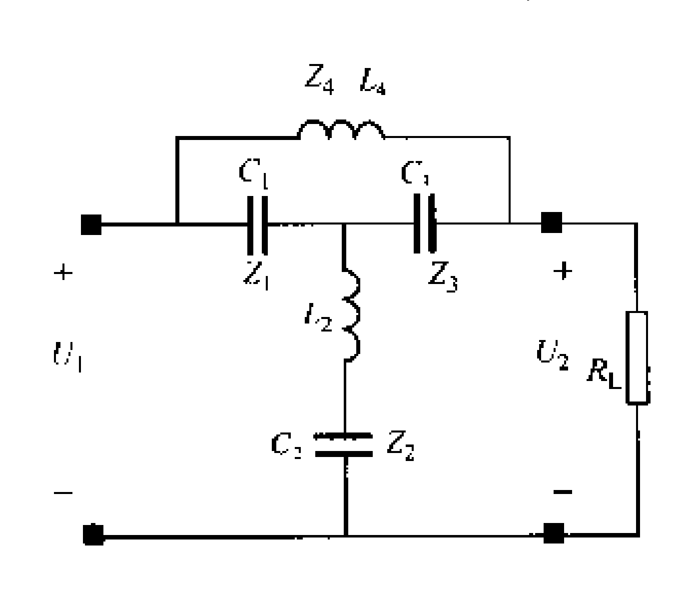


图 5.20-1 原始桥 T 型网络

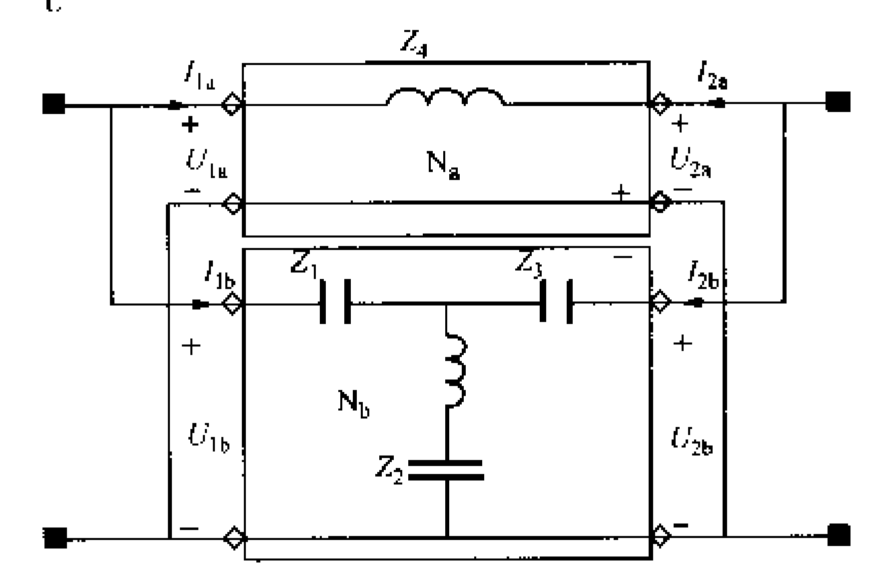


图 5.20-2 等效为网络 N_a 和网络 N_b 并联

于是得桥 T 形电路的 $\boldsymbol{Y}=\boldsymbol{Y}_a+\boldsymbol{Y}_b$，其 $\boldsymbol{Z}$ 矩阵为 $\boldsymbol{Z}=\boldsymbol{Y}^{-1}$

由前知网络函数（本例为电压比函数）

$$H(\mathrm{j}\omega) = \frac{\dot{U}_2}{\dot{U}_1} = \frac{z_{21}Z_L}{\Delta_z + z_{11}Z_L}$$

而输入阻抗为

$$Z_{in} = \frac{\Delta_z + z_{11}Z_L}{z_{22} + Z_L}$$

由此可画出幅频相频响应及输入阻抗随频率变化的曲线。

■ MATLAB 程序 q520.m

```
clear
for k=1:101                              % 对各个频点分别计算其频率特性
   w=0.02*(k-1)+1e-8;              % 为避免奇异值，加一个微量
   C1=2; C3=2; C2=4/3; L2=1; L4=1;         % 元件赋值
   Z1=1./(w*C1*j); Z3=1./(w*C3*j);       % 阻抗计算
   Z2=L2*j*w+1./(j*w*C2);
   Z4=L4*j*w; ZL=1;
   Ya=[1./Z4, -1./Z4; -1./Z4, 1./Z4];            % a 网络参数 Y 计算
   Zb=[Z1+Z2, Z2; Z2, Z3+Z2];      Yb=inv(Zb);            % b 网络参数 Y 计算
   Y=Ya+Yb;     Z=inv(Y);                                 % a, b 网络并联
   H(k)=Z(2, 1)*ZL./(det(Z)+Z(1, 1).*ZL);     % 求 U2/U1
   Zin(k)=(det(Z)+Z(1, 1)*ZL)./(Z(2, 2)+ZL);     % 求输入阻抗
end
w1=[0:100]*0.02;       % 设定绘图横坐标数组 w1, 注意 w 只是一个标量
figure(1)
subplot(2, 1, 1), plot(w1, abs(H))                % 画幅频特性
axis([0 2 0 2])       % 因 abs(H)处处相同，系统无法自动取纵坐标，要人为设定
subplot(2, 1, 2), plot(w1, unwrap(angle(H)))
```

```
% 画相频特性, unwrap 函数用以避免因取主角使小于-pi 的值跳到+pi
axisc[o  2  -10  0]
figure(2)                           % 画网络函数实频特性和虚频特性
subplot(2, 1, 1), plot(w1, real(H))
subplot(2, 1, 2), plot(w1, imag(H))
figure(3)                           % 画输入阻抗的实频特性和虚频特性
subplot(2, 1, 1), plot(w1, real(Zin))
axis([0 2 0 2])                     % 理由同前
subplot(2, 1, 2), plot(w1, imag(Zin))
axisc[0  2  2  0.2]
```

■ **程序运行结果**

见图 5.20-3 和 5.20-4。图 5.20-4 中的尖头是因舍入误差引起的。

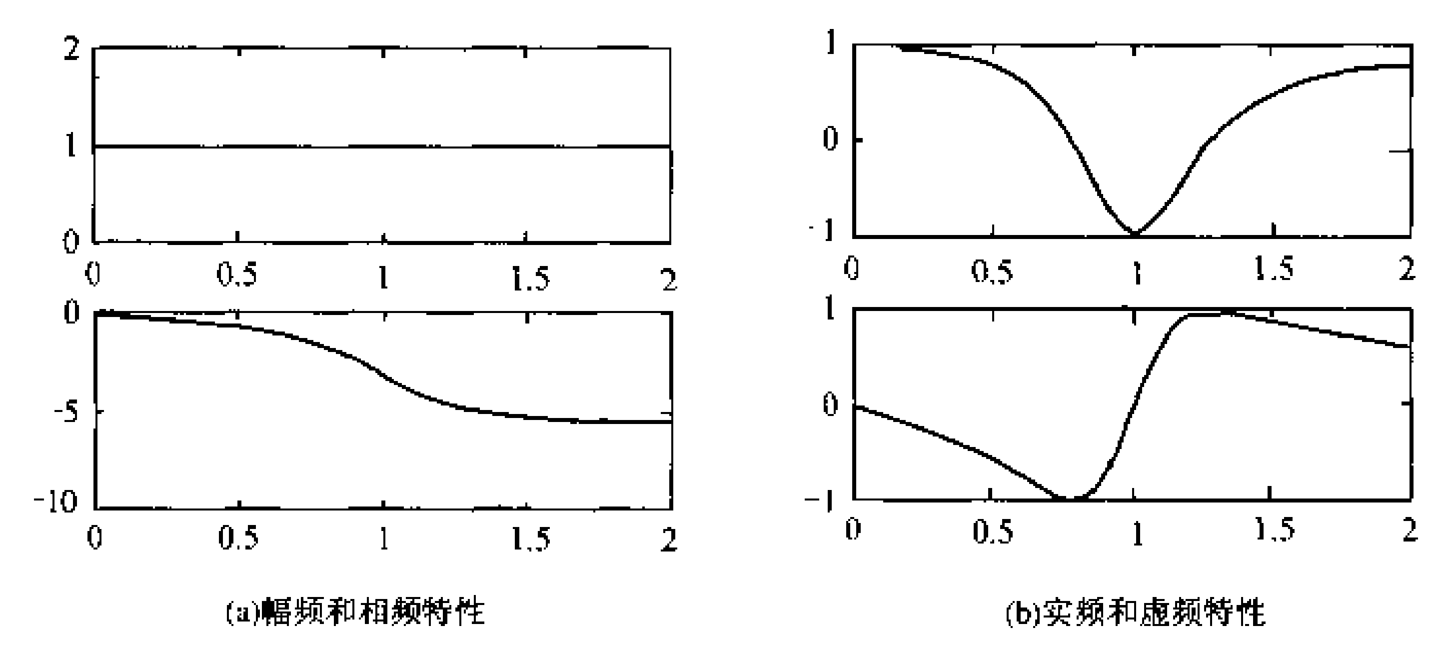


图 5.20-3　网络函数的频率特性

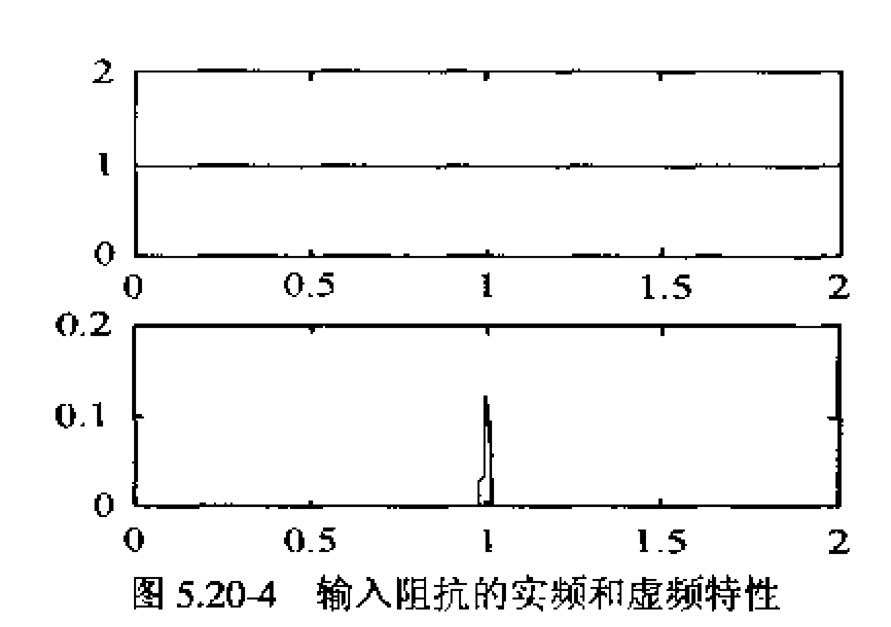


图 5.20-4　输入阻抗的实频和虚频特性

第 6 章　MATLAB 在信号与系统中的应用

MATLAB 特别适用于信号通过系统的分析，本章只介绍一些基本分析计算方法，较深入的内容将在第 7 章、第 8 章讨论。

6.1　连续信号和系统

本节讨论用 MATLAB 表示和分析连续信号和线性时不变（LTI）连续系统的问题。严格说来，只有用符号推理的方法才能分析连续系统。用数值方法是不能表示连续信号的，因为它给出的是各个样本点的数据。只有当样本点取得很密时才可看成连续信号。所谓密，是相对于信号变化的快慢而言。以下均假定相对于采样点密度而言，信号变化足够慢。

【例 6.1】 连续信号的 MATLAB 描述

列出单位冲激函数、单位阶跃、复指数函数等连续信号的 MATLAB 表达式。

解：■ 建模

（1）单位冲激函数 $\delta(t)$无法直接用 MATLAB 描述，可以把它看做是宽度为 Δ（程序中用 dt 表示），幅度为 1/Δ 的矩形脉冲，即

$$X_1(t)=\delta_\Delta(t-t_1)=\begin{cases}\dfrac{1}{\Delta} & t_1<t<t_1+\Delta\\ 0 & \text{其余}\end{cases}$$

表示在 $t=t_1$ 处的冲激。

（2）单位阶跃函数：在 $t=t_1$ 处跃升的阶跃可写为 $u(t-t_1)$。定义为

$$X_2(t)=u(t-t_1)=\begin{cases}1 & t_1<t<t_1+\Delta\\ 0 & t<0\end{cases}$$

（3）复指数函数

$$X_3(t)=\mathrm{e}^{(u+\mathrm{j}\omega)t}$$

若$\omega=0$，它是实指数函数，如 $u=0$，则为虚指数函数，其实部为余弦函数，虚部为正弦函数。本例 $u=-0.5$，$\omega=10$。

■ MATLAB程序q601.m

```
clear,  t0=0; tf=5; dt=0.05; t1=1;
t=[t0: dt: tf];  st=length(t);
% (1)单位冲激信号,
% 在 t1(t0≤t1≤tf)处有一持续时间为 dt, 面积为 1 的脉冲信号, 其余时间均为零.
n1=floor((t1-t0)/dt);        % 求t1对应的样本序号
x1=zeros(1, st);             % 把全部信号先初始化为零
x1(n1)=1/dt;                 % 给出t1处的脉冲信号
subplot(2, 2, 1), stairs(t, x1) ,grid on  %绘图, 注意为何用stairs而不用plot命令
```

```
axis([0, 5, 0, 22])         % 为了使脉冲顶部避开图框，改变图框坐标
%(2)单位阶跃信号，
%信号从t0到tf，在t1(t0<t1<tf) 前为0，到t1处有一跃变，以后为1
%   程序前几句 ，即求t，st，n1的语句与上同，只把x1处改为x2
x2 = [zeros(1, n1-1), ones(1, st-n1+1)];   %产生阶跃信号
subplot(2, 2, 3), stairs(t, x2),grid on              % 绘图
axis([0, 5, 0, 1.1])        % 为了使方波顶部避开图框，改变图框坐标
%(3)复数指数信号
alpha=-0.5;w=10;x3=exp((alpha+j*w)*t);
subplot(2, 2, 2), plot(t, real(x3)),grid on    % 绘图
subplot(2, 2, 4), plot(t, imag(x3)),grid on    % 绘图
```

■ **程序运行结果**

x1，x2，x3 的波形见图 6.1。由于程序中取的 dt 为 0.5s，故在 t=1，$\delta_\Delta(t-1)=\dfrac{1}{\Delta}$=20 处，如果把 dt 设定得更小，脉冲的幅度也将更大。dt 趋近于零时，幅度就趋于无穷大。注意，若要显示连续信号波形中的不连续点，用 stairs 命令；而要使波形光滑些，则用 plot 命令较好。复数指数信号可以分解为余弦和正弦信号，它们分别是复数信号的实部和虚部。图 6.1 右图中的两个衰减振荡信号就代表了这两个相位差 90° 的分量。

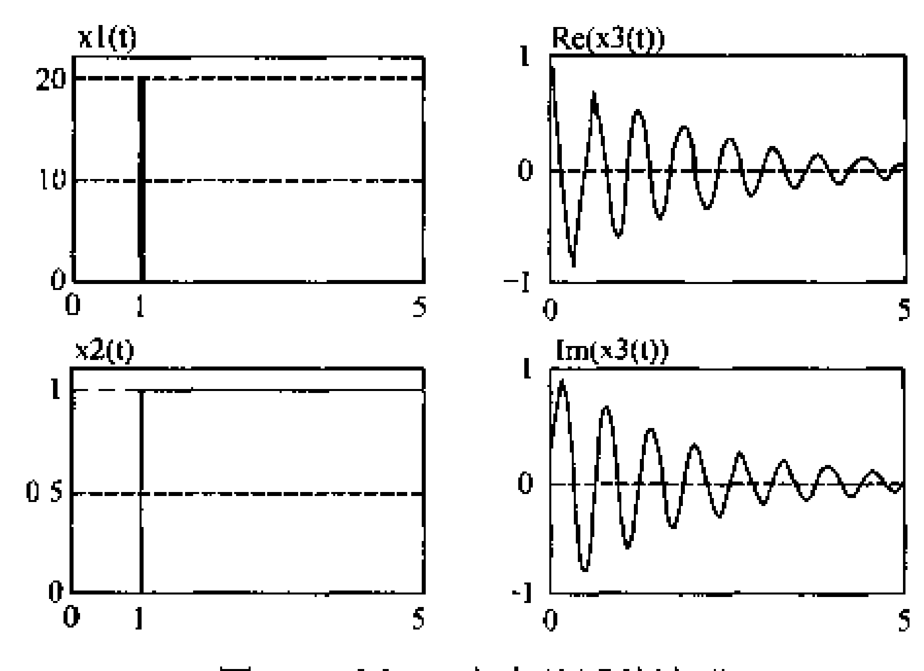


图 6.1　例 6.1 产生的四种波形

【例 6.2】 LTI 系统的零输入响应

描述 n 阶线性时不变（LTI）连续系统的微分方程为

$$a_1\frac{d^n y}{dt^n}+a_2\frac{d^{n-1} y}{dt}+\cdots+a_n\frac{dy}{dt}+a_{n+1}y=b_1\frac{d^m u}{dt^m}+\cdots+b_m\frac{du}{dt}+b_{m+1}u,\qquad n\geqslant m$$

已知 y 及其各阶导数的初始值为 $y(0)$，$y^{(1)}(0)$，…，$y^{(n-1)}(0)$，求系统的零输入响应。

解：■ **建模**

当 LTI 系统的输入为零时，其零输入响应为微分方程的齐次解（即令微分方程等号右端为 0），其形式为（设特征根均为单根）

$$y(t)=C_1e^{p_1t}+C_2e^{p_2t}+\cdots+C_ne^{p_nt}$$

其中 p_1，p_2，…，p_n 是特征方程 $a_1\lambda^n+a_2\lambda^{n-1}+\cdots+a_n\lambda+a_{n+1}=0$ 的根，它们可用 roots(a)语句求得。各系数 C_1，…，C_n 由 y 及其各阶导数的初始值来确定。对此有

$C_1+C_2+\cdots+C_n=y_0$　　　　　$y_0=y(0)$

$p_1C_1+p_2C_2+\cdots+p_nC_n=Dy_0$　　(Dy_0 表示 y 的导数的初始值 $y^{(1)}(0)$)

……………………………………

$p_1^{n-1}C_1+p_2^{n-1}C_2+\cdots+p_n^{n-1}C_n=D^{n-1}y_0$

写成矩阵形式为 $\begin{bmatrix}1 & 1 & \cdots & 1\\ p_1 & p_2 & \cdots & p_n\\ \vdots & \vdots & \ddots & \vdots\\ p_1^{n-1} & p_2^{n-1} & \cdots & p_n^{n-1}\end{bmatrix}\begin{bmatrix}C_1\\ C_2\\ \vdots\\ C_n\end{bmatrix}=\begin{bmatrix}y_0\\ Dy_0\\ \vdots\\ D^{n-1}y_0\end{bmatrix}$

即　　$\boldsymbol{V}\cdot\boldsymbol{C}=\boldsymbol{Y}_0$

其解为　$\boldsymbol{C}=\boldsymbol{V}\backslash\boldsymbol{Y}_0$

式中　　$\boldsymbol{C}=[C_1,C_2,\cdots,C_n]'$; $\boldsymbol{Y}_0=[y_0,Dy_0,\cdots,D^{n-1}y_0]'$

$$\boldsymbol{V}=\begin{bmatrix}1 & 1 & \cdots & 1\\ p_1 & p_2 & \cdots & p_n\\ \vdots & \vdots & \ddots & \vdots\\ p_1^{n-1} & p_2^{n-1} & \cdots & p_n^{n-1}\end{bmatrix}$$

$\boldsymbol{V}$ 为范德蒙矩阵，在 MATLAB 的特殊矩阵库中有 vander。

■ MATLAB程序q602.m

```
a=input('输入分母系数向量a=[a1, a2, ...]= ');
n=length(a)-1;
Y0=input('输入初始条件向量 Y0=[y0, Dy0, D2y0, ...]= ');
p=roots(a);V=rot90(vander(p));c= V\Y0';
dt=input('dt='); tf=input('tf= ') 图 6.2
t=0:dt:tf; y=zeros(1, length(t));
for k=1:n  y= y+c(k)*exp(p(k)*t);end
plot(t, y), grid
```

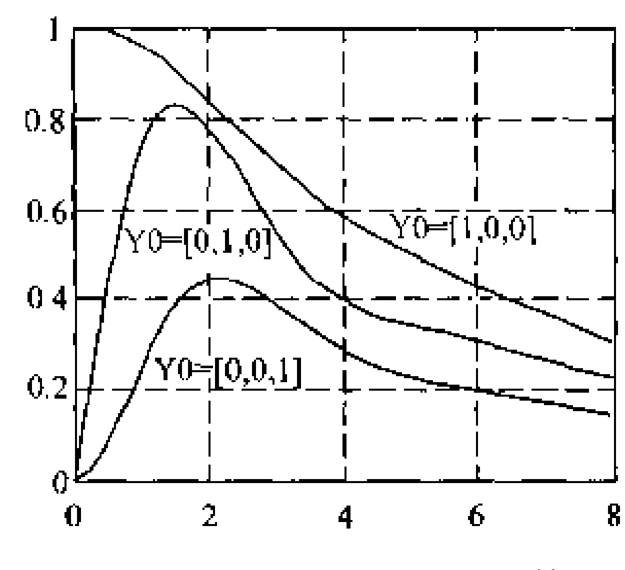


图 6.2　三阶系统零状态分量的解

■ 程序运行结果

用这个通用程序来解一个三阶系统，运行此程序并输入

a=[3, 5, 7, 1];

dt=0.2; tf=8;

而 Y0 取

[1, 0, 0]; [0, 1, 0]; [0, 0, 1]

三种情况，用 hold on 语句使三次运行生成的图形画在一幅图上，得到图 6.2。

【例 6.3】 n 阶 LTI 系统的冲激响应

解：■ 建模

n 阶微分方程如例 6.2 所示，其系统函数为：

$$H(s)=\frac{Y(s)}{U(s)}=\frac{b_1s^m+b_2s^{m-1}+\cdots+b_ms+b_{m+1}}{a_1s^n+a_2s^{n-1}+\cdots a_ns+a_{n+1}}\qquad n\geqslant m$$

其特性可用系统函数分子、分母系数向量 **b** 和 **a** 来表示。

对于物理可实现系统，$n \geqslant m$，即 length(a)≥length(b)。length(a)-1 就是系统的阶次。

冲激函数的拉普拉斯变换为 $U(s)=1$，则系统对冲激函数的响应的拉普拉斯变换为 $Y(s) = H(s)U(s) = H(s)$，冲击响应就是 $H(s)$的拉普拉斯反变换，可以把 $H(s)$展开为极点留数式。如果 $H(s)$的分母多项式没有重根，则

$$H(s)=\sum_{k=1}^{n}\frac{r_k}{s-p_k}$$

故有

$$h(t)=\sum_{k=1}^{n}r_k e^{p_k t}$$

■ **MATLAB程序q603.m**

```
a=input('多项式分母系数向量a=  ')
b=input('多项式分子系数向量b=  ')
[r, p] = residue(b,  a),             % 求极点和留数
disp('解析式h(t)=Σr(i)*exp(p(i)*t)')
disp('给出时间数组t=[0:dt:tf]')
dt=input('dt= ');  tf=input('tf= '); % 输入dt及终点tf
t=0:dt:tf;                     %  给定时间数组
h=zeros(1, length(t));  %  h的初始化
for  i=1:length(a)-1                 % 根数为a的长度减1
   h = h+ r(i)*exp(p(i)*t);          % 叠加各根分量
end
plot(t, h), grid
```

■ **程序运行结果**

用通用程序求一个五阶系统的冲激响应。按提示输入分子分母系数向量和时间数组。

```
b=[8, 3, 1];
a= poly([0, -1+2i, -1-2i, -2, -5]);
t=0:0.2:8;
```

因为题中给出的分母是系统的极点，而不是多项式系数，要求出其系数可用 poly 函数，其格式为 a=poly(p)（其中 p 为极点向量），即可求出 h，画出的曲线如图 6.3 所示。

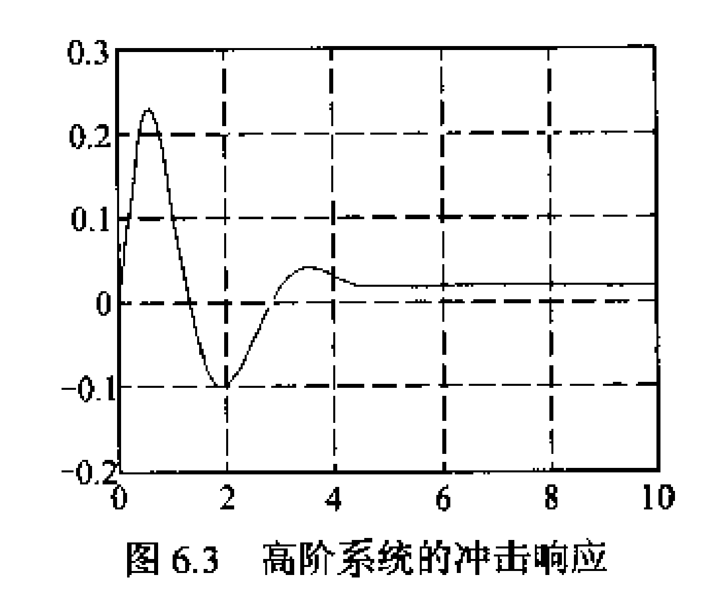


图 6.3　高阶系统的冲击响应

【例 6.4】 卷积的计算

某 LTI 系统的冲激响应 $h(t)=e^{-0.1t}$，输入 $u(t)$ 如图 6.4(a)所示，初始条件为零，求系统的响应 $y(t)$。

解：■ 建模

根据卷积公式：

$$y(t)=\int_0^t u(\tau)h(t-\tau)\mathrm{d}\tau$$

MATLAB 的基本函数中，有卷积函数 conv，可以直接调用它。因此编程的过程为：

（1）写出 $h(t)$的 MATLAB 表达式；

（2）写出 $u(t)$的 MATLAB 表达式；

（3）利用 MATLAB 的卷积语句 y=conv(u，h)求解并画曲线。

■ MATLAB程序q604a. m及q604b. m

```
% （1）q604a.m，调用conv函数的简单程序
u=input('输入u数组 u= ');          % 输入u，h序列
h=input('输入h数组 h= ');
dt=input('输入时间间隔dt= ');
y =conv(u, h);                      % 调用conv函数
% 在下面的语句中，注意不知t的长度而要用它绘图时， t数组的设定方法
plot(dt* ([1:length(y)]-1), y), grid
% （2）q604b.m，自编卷积函数并作动态演示的复杂程序
% 本程序为读者深入理解和教师讲解卷积过程提供了演示工具
clear
u1s=input('输入u数组 u=（例如ones(1, 10)）  ');
lu=length(u1s);
h1s=input('输入h数组 h=（例如exp(-0.1* [1:15])）  ');
lh=length(h1s);
lmax=max(lu, lh);
if lu>lh nu=0; nh=lu-lh;           % 若u比h长，对h补nh个零
elseif lu<lh nh=0; nu=lh-lu;       % 若h比u长，对u补nu个零
else nu=0; lh=0;                    % 若h与u同长，不补零
end
dt=input('输入时间间隔dt=（例如0.5）  ');
lt=lmax;                           % 取长者为补零长度基准
% 将u先补得与h同长，再两边补以同长度的零
u=[zeros(1, lt), u1s, zeros(1, nu), zeros(1, lt)];
t1=(-lt+1:2*lt) *dt;
% 将h先补得与u同长，再两边补以同长度的零
h=[zeros(1, 2*lt), h1s, zeros(1, nh)];
hf=fliplr(h);                      % 将h的左右翻转，称为hf
y=zeros(1, 3*lt);                  % y长度初始化
for k=0:2*lt
   p=[zeros(1, k), hf(1:end-k)];          % 使hf向右循环移位
   y1=u. *p*dt;           % 使输入和翻转移位的脉冲过渡函数逐项相乘，再乘dt
   yk=sum(y1);            % 相加，相当于积分
   y(k+lt+1)=yk;                    % 将结果放入数组y
% 绘图，注意如何用axis命令把各子图的横坐标统一起来，纵坐标随数据自动调整
   subplot(4, 1, 1); stairs(t1, u)
% 用stairs是为了避免plot函数在突跳点形成的斜边
```

```
    axis([-lt*dt, 2*lt*dt, min(u), max(u)]), hold on
    ylabel('u(t)')
    subplot(4, 1, 2);stairs(tl, p)
    axis([-lt*dt, 2*lt*dt, min(p), max(p)])
    ylabel('h(k-t)')
    subplot(4, 1, 3);stairs(tl, y1)
    axis([-lt*dt, 2*lt*dt, min(y1), max(y1)+eps])
    ylabel('s=u*h(k-t)')
    subplot(4, 1, 4);stem(k*dt, yk)          % 用stem函数表示每一次卷积求和的结果
    axis([-lt*dt, 2*lt*dt, floor(min(y)+eps), ceil(max(y+eps))])
    hold on,  ylabel('y(k)=sum(s) *dt')
    if k==round(0.8*lt)  disp('暂停，按任意键继续'),  pause
    else pause(1),
    end
end
```

■ **程序运行结果**

分别运行这两个程序，按提示输入以下数据

u 数组：　u= ones(1，10)

h 数组：　h= exp(-0.1*[1:15])

时间间隔：　dt= 0.5

得到的图形如图 6.4 所示。这个图形是在卷积进行到一多半时的记录。

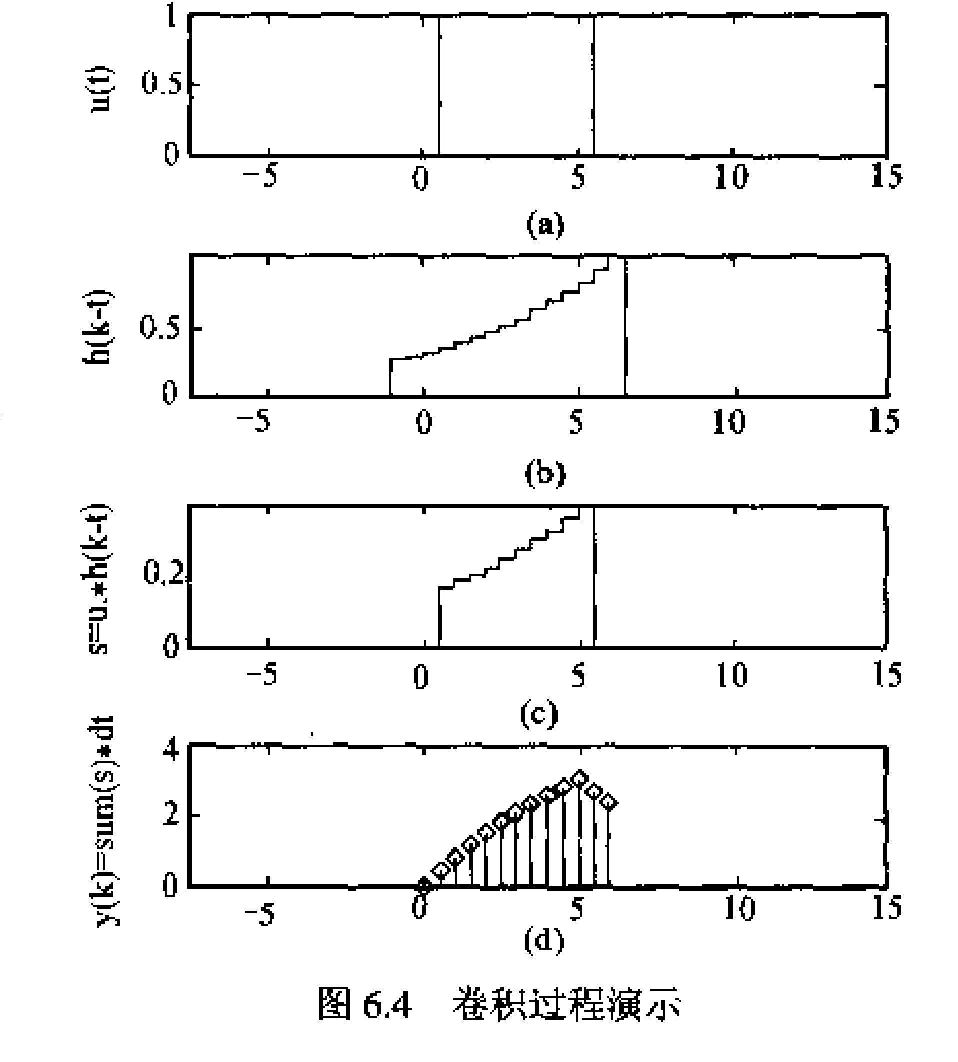


图 6.4　卷积过程演示

程序 q604a.m 只能得到最下面的输出图，因为 u，h，y 三个变量的长度不同，而且时间基准在每一次卷积时不同，很难把它们对照起来画图。要描述卷积过程和它们的时间关系，就要如 q604b.m 那样自己编程。

【例 6.5】 LTI 系统的零状态响应

设二阶连续系统，其特性可用常微分方程表示

$$\frac{d^2y}{dt^2}+2\frac{dy}{dt}+8y=u$$

求其冲激响应。若输入为 $u=3t+\cos(0.1t)$，求其零状态响应 $y(t)$。

解：■ **建模**

先求系统的冲激响应，写出其特征方程

$$\lambda^2+2\lambda+8=0$$

按例 6.3 求出其特征根为 p_1，p_2，及相应的留数 r_1，r_2，则冲击响应为

$$h(t)=r_1e^{p_1t}+r_2e^{p_2t}$$

输出 $y(t)$可用输入 $u(t)$与冲击响应 $h(t)$的卷积求得。

■ MATLAB程序q605.m

```
clf, clear
a=input('多项式分母系数向量 a= ');
b=input('多项式分子系数向量 b= ');
t=input('输入时间序列 t=[0:dt:tf] ');
u=input('输入序列 u=  ');
tf=t(end);
dt=tf/(length(t)-1);
%用极点留数法求冲注激响应
[r, p, k]=residue(b, a);
h=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t);
% 画出冲击响应 h(t)
subplot(2, 1, 1), plot(t, h);grid
% 求 u 和 h 的卷积, 得输出 y(t)
y=conv(u, h)*dt;
% 画出输出 y(t)
subplot(2, 1, 2),
plot(t, y(1:length(t)));grid      % 画出输出 y(t)
```

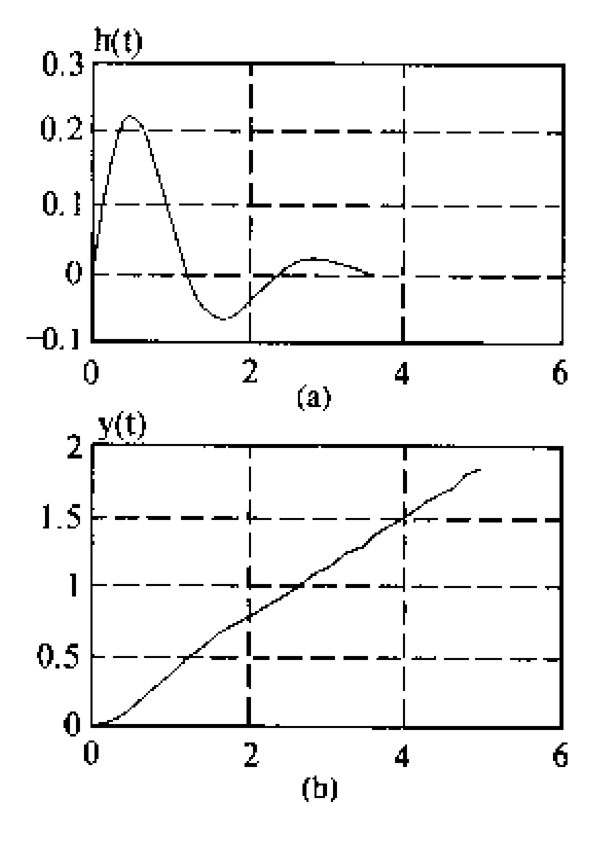


图 6.5　冲击响应和卷积法求输出

■ **程序运行结果**

执行这个程序，取 a= [1，2，8]，b=1，t=[0:0.1:5]及 u=3*t+cos(0.1*t)，所得的结果见图 6.5。图 6.5(a)显示冲击响应 h。

最后两行程序用来求给定输入下的输出，这里用卷积函数来求，即线性系统的输出等于输入信号和系统冲激响应的卷积。注意卷积求得的输出序列长度为 u 和 h 长度之和减 1。在程序执行后用 whos 命令检查，可知 y 的长度为 101，而 t 的长度为 51，故最后一句绘图命令必须限定 y 中的前面 51 个点。读者可自行分析 y 的后 50 个点具有何种意义。

【例 6.6】 系统中有重极点时的计算

n 级放大器，每级的传递函数均为$\omega_0/(s+\omega_0)$，求阶跃响应，画出 n 不同时的波形和频率特性。

解：■ 建模

系统的转移函数为 $H(s)=\omega_0^n/(s+\omega_0)^n$，阶跃信号的拉普拉斯变换为 $1/s$，因此，输出为两者的乘积，即

$$Y(s)=\frac{\omega_0^n}{s(s+\omega_0)^n}$$

求 $Y(s)$的拉普拉斯反变换，即可得到阶跃响应 $y(t)$。

这里遇到了有多重极点$-\omega_0$的 $H(s)$，求其拉普拉斯反变换的问题。依照这个问题的原理，也可以用留数极点分解法来求，不过在有 n 重根时分解出的部分分式的分母将是重极点，有$(s+\omega_0)$，…，$(s+\omega_0)^{n-1}$，$(s+\omega_0)^n$等项，$(s+\omega_0)^{-q}$的反变换可用如下解析式表示

$$L^{-1}\left[\left(\frac{1}{s+\omega_0}\right)^q\right]=\frac{1}{(q-1)!}t^{q-1}e^{-\omega_0 t}$$

按照这个思路，应该先用下列语句求 $Y(s)$的极点留数，注意分母中除了有 n 个重极点外还有一个零极点（即 $1/s$），故共有 $n+1$ 个极点。先用 poly 函数求 $Y(s)$分母多项式的系数向量

by = w0^n; ay=[poly(-ones(1，n)*w0)，0]

后，放在 residue 的输入参数中。

[r，p] = residue(by，ay);

然后，检查其中的重极点的个数，对非重极点用公式 r(k)*exp(p(k)*t)，而对重极点则用公式 r(k)*t.^(k-1).*exp(p(k)*t)/prod(1: k-1)。求出其分量，再把各个分量叠加起来。

实际上，这样编程不仅非常麻烦，而且难以算出正确的结果。其原因是在重极点处，MATLAB 的 residue 算法遇到了病态问题，数据中小小的舍入误差可能造成结果出现很大的误差。即使 n 取 2 都得不出正确结果。

为了避开重极点问题，可以有意把极点拉开一些，例如设 n 个极点散布在$-0.95\omega_0$到$1.05\omega_0$之间，那样也就可当非重极点来列程序。这种处理在工程上是完全没问题的，一般电阻的标准误差为±5%，电容则更大，使各个放大器常数完全相同是不可能的，要把其误差控制到±2%以内也非易事。由此可用非重极点的程序来求输出。

■ **MATLAB程序q606.m**

```
clear, clf,  N=input('输入放大器级数 N=');
w0=1000; dt=1e-4; tf=0.01; t = 0:dt:tf;
y=zeros(N, length(t));                  % 输出初始化
for n=1:N
   p0=-linspace(.95, 1.05, n)*w0; % 将 H(s)极点分散布置
     ay = poly([p0, 0]);        % 由 Y(s)的极点（比 H(s)多一个零极点）求分母系数
     by = prod(abs(p0));        % 求 Y(s)的分子系数
   [r, p] = residue(by, ay);       % 求 Y(s)的极点留数
   for k = 1: n+1                          % 把各部分分式对应的时域分量相加
      y(n, :) = y(n, :) + r(k)*exp(p(k)*t);
   end
   figure(1), plot(t,  y(n, : ));grid on, hold on          % 绘制过渡过程曲线
%下面这几条语句用来绘制波德 bode 图，如果用控制系统工具箱中的 bode 函数，只要一句即可
%    figure(2), bode(prod(abs(p0)),  poly(p0));hold on
bh=by;ah= poly(p0);              % 求 H(s)的分子分母系数
w=logspace(2, 4);                    % 给出频率范围和分度
H = polyval(bh, j*w)./polyval(ah, j*w); % 求 H(s)在各频点的值 H(jw)
aH=unwrap(angle(H))*180/pi;        % 求出以度为单位的连续的相角
```

```
fH=20*log10(abs(H));                          % 求出以分贝为单位的振幅
figure(2),
subplot(2, 1, 1), semilogx(w, fH), grid on, hold on % 绘幅频图
subplot(2, 1, 2), semilogx(w, aH), grid on, hold on % 绘相频图
end,  hold off
```

■ **程序运行结果**

设 N=4，可得过渡过程如图 6.6-1，从中看出输出信号达到 0.6 处所需的时间约为单级时常数乘以级数。此程序在 N>4 时又会出现很大误差。

为了画波德图，要把坐标设置得符合它的要求。下面是画波德图的几项说明。

(1) 求频率特性，用多项式求值函数 polyval，并且用了元素群运算，把频率数组作为自变量，一次就求出全部的频率特性。注意它们是复数，要同时关心它们的振幅和相角；

(2) 振幅和相位特性横坐标都要用对数坐标并且上下对齐；

(3) 振幅纵坐标单位应为分贝，这里用了 20*log10(abs(H))；

(4) 相角纵坐标单位为度，并且应连续变化，所以这里加了 unwrap 命令。

在控制系统工具箱中只要一个 bode 命令就可以完成这些功能，但这里不用工具箱，以便让读者知道工具箱是怎么编程的，今后调用时，不但知其然，而且知其所以然。

图 6.6-2 绘出了多级放大器的频率特性，其中幅特性（图上为分贝数）显示了低通特性，随级数的增加，通带减小；从相特性看出，随级数的增加，负相移成比例地增加。

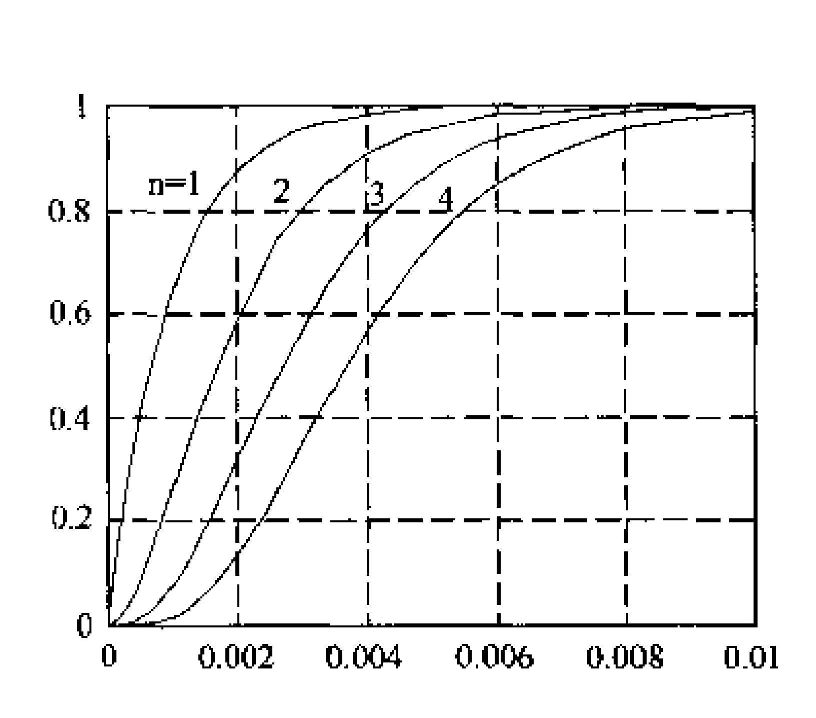


图 6.6-1　多级放大器的阶跃过渡过程

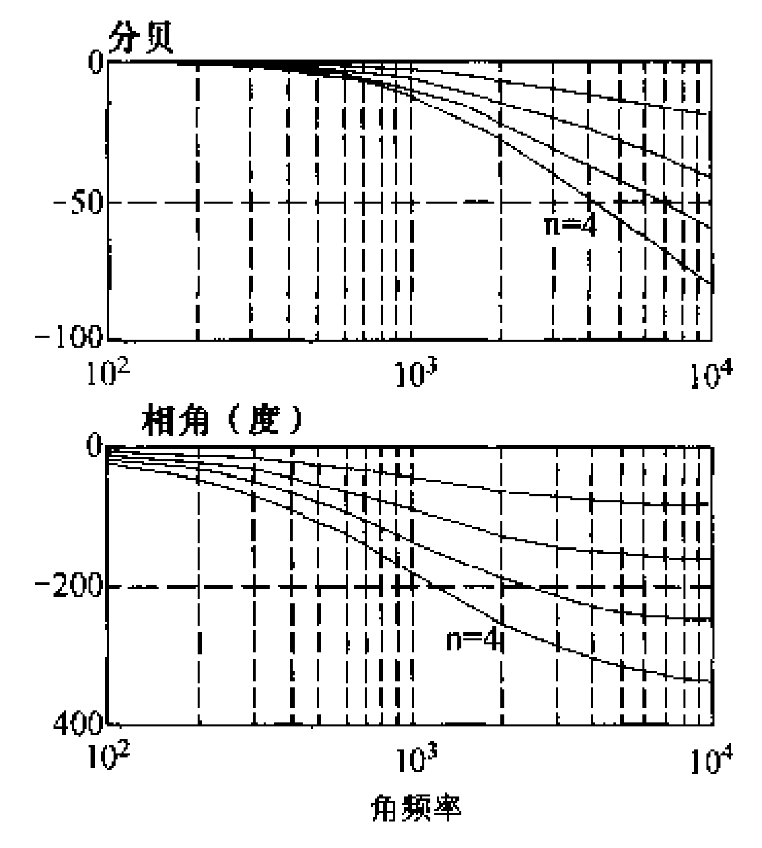


图 6.6-2　多级放大器的频率特性

6.2　傅立叶分析

【**例 6.7**】 方波分解为多次正弦波之和

如图 6.7-1 所示的周期性方波，其傅立叶级数为

$$f(t)=\frac{4}{\pi}\left[\sin t+\frac{1}{3}\sin 3t+\cdots+\frac{1}{2k-1}\sin(2k-1)t+\cdots\right]\qquad k=1，2，\cdots$$

用 MATLAB 演示谐波合成情况。

解：■ 建模

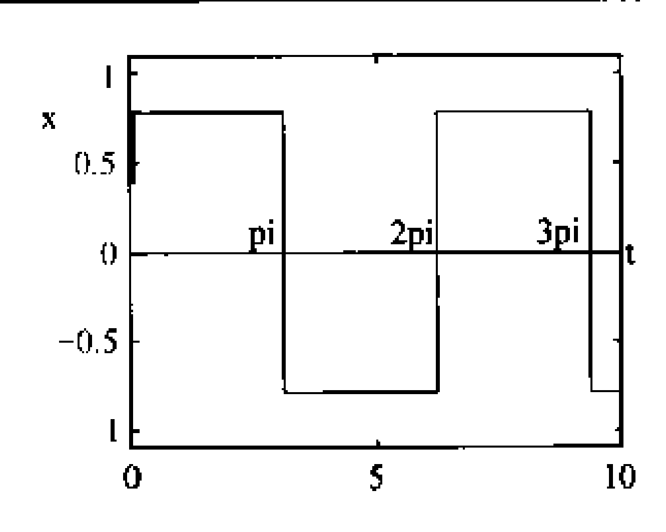


图 6.7-1 输入周期性方波

方波 $f(t)$ 的周期 $T=2\pi$，由于该方波是奇对称的，在 $t=0\sim\pi$ 间演示即可，分别计算

$$f_1(t)=\frac{4}{\pi}\sin t$$

$$f_3(t)=\frac{4}{\pi}(\sin t+\frac{1}{3}\sin 3t)$$

……

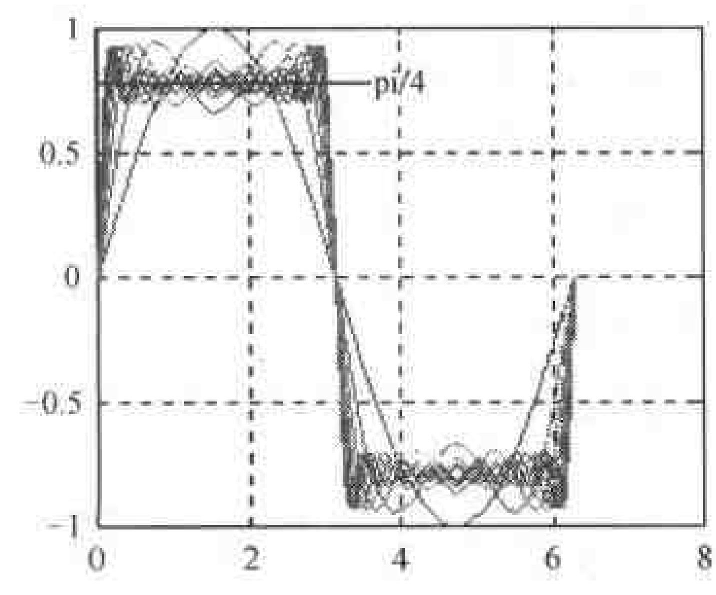
直到9次谐波，并做图。

■ MATLAB程序q607.m

```
t = 0:.01:2*pi;                          % 设定一个时间数组，有101个点
y = sin(t);plot(t, y), figure(gcf), pause       % 频率为w=1(f=1/2π)的基波
y = sin(t) + sin(3*t)/3; plot(t, y),  pause     %叠加三次谐波
%用1, 3, 5, 7, 9次谐波叠加
y=sin(t) + sin(3*t)/3 + sin(5*t)/5 + sin(7*t)/7 + sin(9*t)/9;
plot(t, y)
%为了绘制三维曲面，要把各次波形数据存为一个三维数组，因此必须重新定义y
y = zeros(10, max(size(t))); x = zeros(size(t));
for k=1:2:19
x = x + sin(k*t)/k; y((k+1)/2, : ) = x;
end
%将各波形叠合绘出
pause,  figure(1), plot(t, y(1:9, : )), grid
line([0, pi+0.5], [pi/4, pi/4])                     % 加上方波幅度线及标注
text(pi+0.5, pi/4, 'pi/4')
%将各半波形绘成三维网格图，看出增加谐波阶次对方波逼近程度的影响
halft=ceil(length(t)/2); pause,
figure(2), mesh(t(1:halft), [1:10], y(:, 1:halft))  % 只用正半周波形
```

■ 程序运行结果

如图 6.7-2 所示。



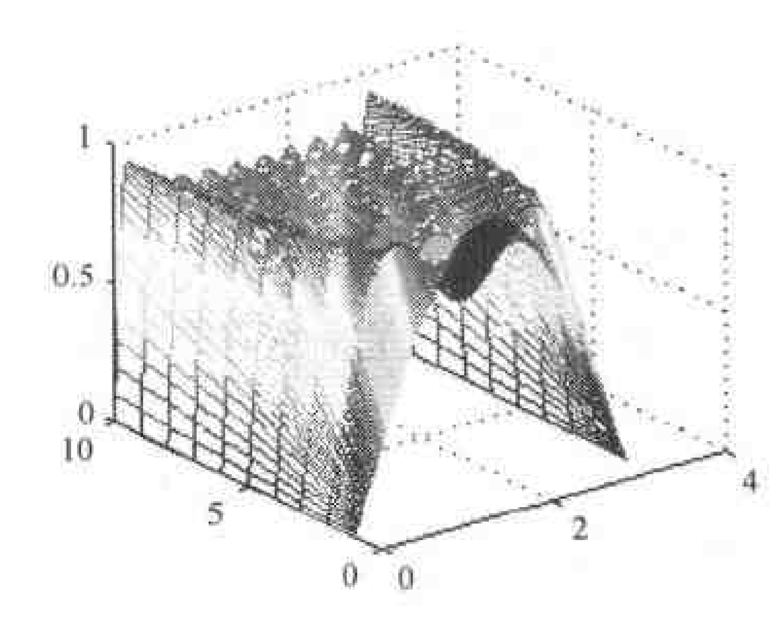


图 6.7-2 方波分解为正弦波

【例 6.8】 周期信号的频谱

周期电流、电压（统称其为信号）$f(t)$可展开为直流与各次谐波之和，即

$$f(t)=\frac{A_0}{2}+\sum_{k=1}^{\infty}A_{km}\cos(k\Omega t+\varphi_k)$$

式中$\Omega=\dfrac{2\pi}{T}$是基波角频率，T为周期。

$$A_{km}=\sqrt{a_k^{\ 2}+b_k^{\ 2}}$$

$$\alpha_k=\frac{2}{T}\int_{-T/2}^{T/2}f(t)\cos(k\Omega t)\mathrm{d}t\,,\qquad k=0,\ 1,\ 2,\ \cdots \tag{6.1}$$

$$b_k=\frac{2}{T}\int_{-T/2}^{T/2}f(t)\sin(k\Omega t)\mathrm{d}t\,,\qquad k=0,\ 1,\ 2,\ \cdots$$

周期信号的有效值定义为

$$A=\sqrt{\frac{1}{T}\int_0^T[f(t)]^2\mathrm{d}t} \tag{6.2}$$

若用各谐波有效值$\left(\dfrac{1}{\sqrt{2}}A_{km}\right)$，则表示为

$$A=\sqrt{A_0^2+\sum_{k=1}^{\infty}\left(\frac{1}{\sqrt{2}}A_{km}\right)^2} \tag{6.3}$$

全波整流电压$U_s(t)$的波形如图 6.8-1 所示。用傅立叶级数可求得

$$A_0=\frac{4U_m}{\pi},\quad A_{km}=\frac{1}{k^2-1}\frac{4U_m}{\pi}\qquad k=2,\ 4,\ 6,\ \cdots$$

$$A_{km}=0\qquad k=1,\ 3,\ 5,\ \cdots$$

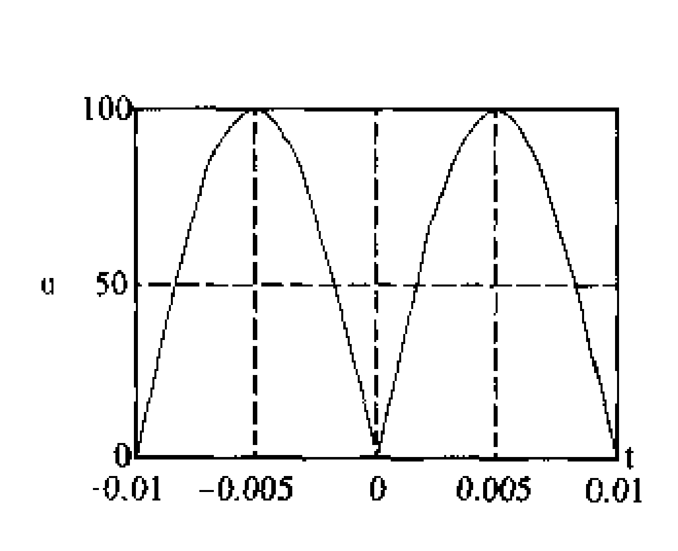


图 6.8-1　半波信号的波形图

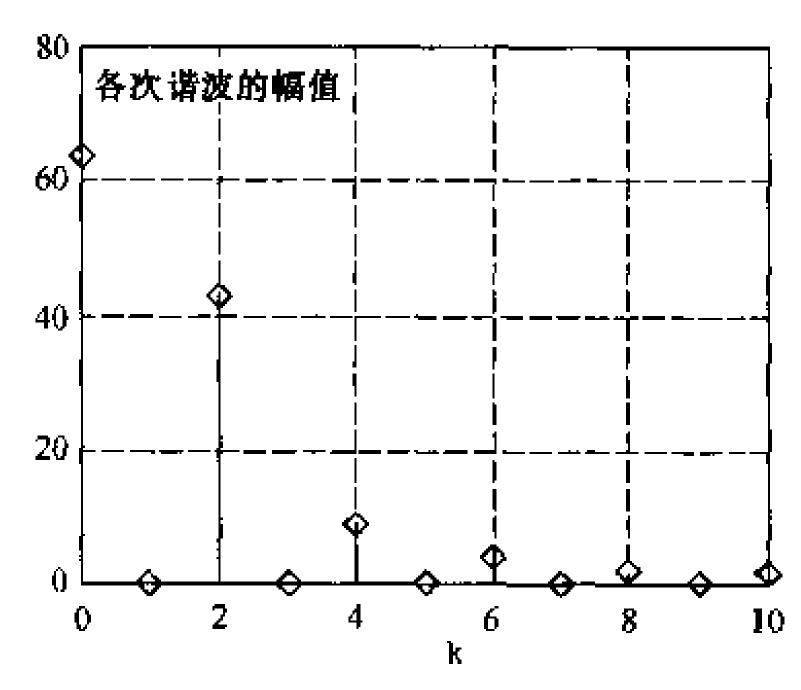


图 6.8-2　半波信号的各谐波分量

可写出其展开式为（它只含直流和偶次谐波，令$k=2n$）

$$u_s(t)=\frac{4U_m}{\pi}\left[\frac{1}{2}-\sum_{n=1}^{\infty}\frac{1}{4n^2-1}\cos 2n\Omega t\right]$$

若U_m=100V，频率$f=50$Hz（相应的T=0.02s，$\omega_1=100\pi$ rad/s)，分别用式（6.1）和式（6.2）计算其有效值U_{s_1}和U_{s_2}（取至 6 次谐波），并求U_{s_2}的误差。

解：■ 建模

令 $\theta=\Omega t$ ，$u_s(t)=\left|U_m\sin\Omega t\right|=\left|U_m\sin\theta\right|$，则

$$U_{s_1}=\sqrt{\frac{1}{T}\int_0^T\left[U_m\sin\Omega t\right]^2\mathrm{d}t}=U_m\sqrt{\frac{1}{2\pi}\int_0^{2\pi}\sin^2\theta\,\mathrm{d}\theta}=U_m\sqrt{\frac{1}{\pi}\int_0^{\pi}\sin^2\theta\,\mathrm{d}\theta} \tag{6.4}$$

由前 n 项分量的功率求出的有效值为

$$U_{s_2}=\frac{4U_m}{\pi}\sqrt{\left(\frac{1}{2}\right)^2+\frac{1}{2}\sum_{n=1}^{3}\left(\frac{1}{4n^2-1}\right)^2}$$

U_{s_2}的误差为 $\varepsilon=\dfrac{U_{s_2}-U_{s_1}}{U_{s_1}}$。

■ MATLAB程序q608.m

```
clear,  format compact
Um=100; T=0.02; w=2*pi*50;
% 方法 1: 按傅立叶分析定义计算
N=input('取的谐波次数 N=  ');        % 取的谐波次数愈高，分段数愈多
t=linspace(-T/2, T/2);dt=T/99;       % 列出 t 数组，取 100 点，有 99 个间隔 dt
u=Um*abs(sin(w*t));              % 在一个周期内生成两个半波的简便方法
for k=0:N                            % 循环求系数 ak, bk, Ak
   a(k+1)=trapz(u.*cos(k*w*t))*dt/T*2;      % 变量下标不得取 0，故将 k 加 1
   b(k+1)=trapz(u.*sin(k*w*t))*dt/T*2;
   A(k+1)=sqrt(a(k+1)^2+b(k+1)^2);
end
[[0:N]', [A(1)/2, A(2:end)]']            % 显示各傅立叶分量，恢复与 k 的对应关系
stem(0:N, [A(1)/2, A(2:end)])            % 画出各傅立叶分量与 k 的对应关系
Us11=sqrt(trapz(u.^2) *dt/T)       % 用总功率求出的有效值
Us12=sqrt(A(1)^2/4+sum(A(2:end).^2/2))      % 用各分量功率和求出的有效值
% 方法 2: 按推导出的全波傅立叶分量公式计算
Us21=Um*sqrt(trapz(sin(w*t).^2)*dt/T)            % 用总功率求出的有效值
% 用前四个分量功率和求出的有效值
Us22=4*Um/pi*sqrt(0.5^2+0.5*sum((1./(4* [1:3].^2-1)).^2))
e=(Us21-Us22)/Us21                   % 相对误差
```

■ 程序运行结果

运行程序，按提示输入。

取得谐波次数 N=10

```
ans =    0          63.6566
         1.0000      0.0321
         2.0000     42.4520
         3.0000      0.0321
         4.0000      8.4990
         5.0000      0.0323
```

```
     6.0000     3.6485
     7.0000     0.0325
     8.0000     2.0317
     9.0000     0.0327
    10.0000     1.2968
```

■ 方法 1 的计算结果

Us11 =　70.7107（由总功率求出的有效值）

Us12 =　70.7031 [这是取 10 次谐波（也即前六个分量）的功率求出的有效值]

■ 方法 2 的计算结果

Us21 =　70.7107（由总功率求出的有效值）

Us22 =　70.6833 [这是取 6 次谐波（也即前三个分量）的功率求出的有效值]

e =　3.8785e-004

可以看到计算结果中的奇次谐波虽然很小，但不为零。这是由于数值积分误差造成的。如果把步长减小，将 $-T/2$ 与 $T/2$ 之间分为 1000 点。即修改程序中的语句为：

t=linspace(-T / 2，T / 2，1000); dt=T / 999;

运行得出的一次谐波就为 0.000314，减小了 100 倍。其他奇次谐波也相仿。

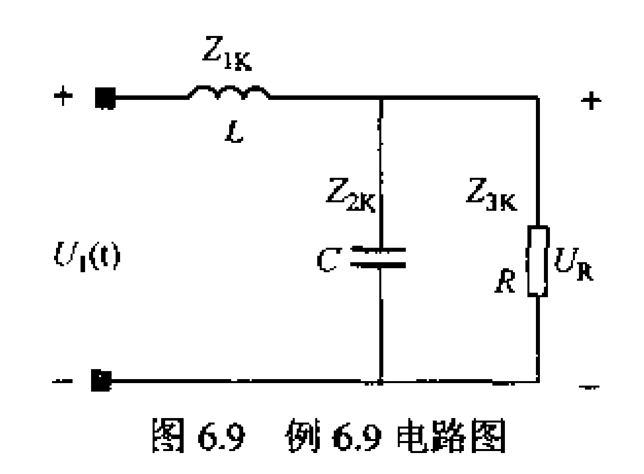


图 6.9　例 6.9 电路图

【例 6.9】 周期信号的滤波

如图 6.9 滤波电路，已知 L=400mH，C=10μF，R_1=200Ω。如激励电压 $u_s(t)$ 为全波整流信号，U_m=100V，ω_1=100πrad/s，求负载 R 两端的直流和各次谐波（它只含偶次谐波）分量。

解：■ 建模

$$u_s(t)=\frac{4U_m}{\pi}\left[\frac{1}{2}-\sum_{n=1}^{\infty}\frac{1}{4n^2-1}\cos 2n\omega_1 t\right]$$

由于直流电感阻抗为 0，电容开路，故

$$U_{R_0}=\frac{2U_m}{\pi}$$

对于第 k 次谐波电压相量

$$\dot{U}_{R_k}=\frac{\dfrac{Z_{2k}Z_{3k}}{Z_{2k}+Z_{3k}}}{Z_{1k}+\dfrac{Z_{2k}Z_{3k}}{Z_{2k}+Z_{3k}}}\dot{U}_{s_k}=\frac{Z_{2k}Z_{3k}}{Z_{1k}Z_{2k}+Z_{2k}Z_{3k}+Z_{1k}Z_{3k}}U_{s_k}$$

式中
$$\dot{U}_{s_k}=\frac{1}{k^2-1}=\frac{1}{4n^2-1}\qquad (k=2n,\ n=1,2,3,)$$

为电源 k 次谐波相量。

$$Z_{1k}=\mathrm{j}k\omega_1 L=\mathrm{j}2n\omega_1 L$$

$$Z_{2k}=-\mathrm{j}\frac{1}{k\omega_1 C}=-\mathrm{j}\frac{1}{2n\omega_1 C}$$

$$Z_{3k}=R$$

$$\dot{U}_{R_k}=\frac{-\mathrm{j}\dfrac{R}{k\omega_1 C}\cdot\dot{U}_{s_k}}{\dfrac{L}{C}+\mathrm{j}\left(k\omega_1 LR-\dfrac{R}{k\omega_1 C}\right)}$$

由此式可求得U_R的各次谐波。

■ **MATLAB程序q609.m**

```
clear, format compact
L=0.4;C=10e-6;R=200;                         % 输入元件参数
Um=100;w1=100*pi;
% 输入只有偶次谐波，故只分析偶次谐波
N=input('需分析的谐波次数 2N=（键入偶数）　');
n=1:N/2;w=[eps, 2*n*w1];                          % 设定频率数组
Us = 4*Um/pi* [0.5, -1./(4*n.^2-1)];  % 输入信号频谱数组
z1=j*w*L;z2=1./(j*w*C);z3=R;        % 计算阻抗数组
z23=z2.*z3./(z2+z3);                % 注意为何用元素群运算符
Ur=Us.*z23./(z1+z23)                % 求 UR
disp('  谐波次数    谐波幅度   谐波相移(度)')
disp([2* [0, n]', abs(UR)', angle(UR)'*180/pi])
```

■ **程序运行结果**

根据程序提示：需分析的谐波次数 2N=（键入偶数）

键入 10 后，得到

```
UR=
  Columns 1 through 4
  63.6620 -0.0000i  12.8383 +27.8571i   1.3050 + 0.6169i   0.2546 + 0.0726i
  Columns 5 through 6
   0.0799 + 0.0165i   0.0326 + 0.0053i
  谐波次数       谐波幅度       谐波相移(度)
     0           63.6620          0.0000
     2.0000      30.6731         65.2568
     4.0000       1.4434         25.3014
     6.0000       0.2648         15.9253
     8.0000       0.0816         11.7029
    10.0000       0.0330          9.2740
```

【例 6.10】 调幅信号通过带通滤波器

已知带通滤波器的系统函数为

$$H(s)=\frac{U_2(s)}{U_1(s)}=\frac{2s}{(s+1)^2+100^2}$$

激励电压 $u_1(t)=(1+\cos t)\cos(100t)$

求（1）带通滤器的频率响应;

（2）输出的稳态响应 $u_2(t)$并画出波形。

解：■ 建模

用傅立叶级数激励信号 $u_1(t)$可展开为

$$u_1(t)=\frac{1}{2}\cos(99t)+\cos(100t)+\frac{1}{2}\cos(101t)$$

即各相量为 $\dot{U}_1(99)=\frac{1}{2}\angle 0^\circ$，$\dot{U}_1(100)=1\angle 0^\circ$，$\dot{U}_1(101)=\frac{1}{2}\angle 0^\circ$

带通滤波器的频率响应

$$H(\mathrm{j}\omega)=H(s)\Big|_{s=\mathrm{j}\omega}=\frac{\mathrm{j}2\omega}{(\mathrm{j}\omega+1)^2+100^2}$$

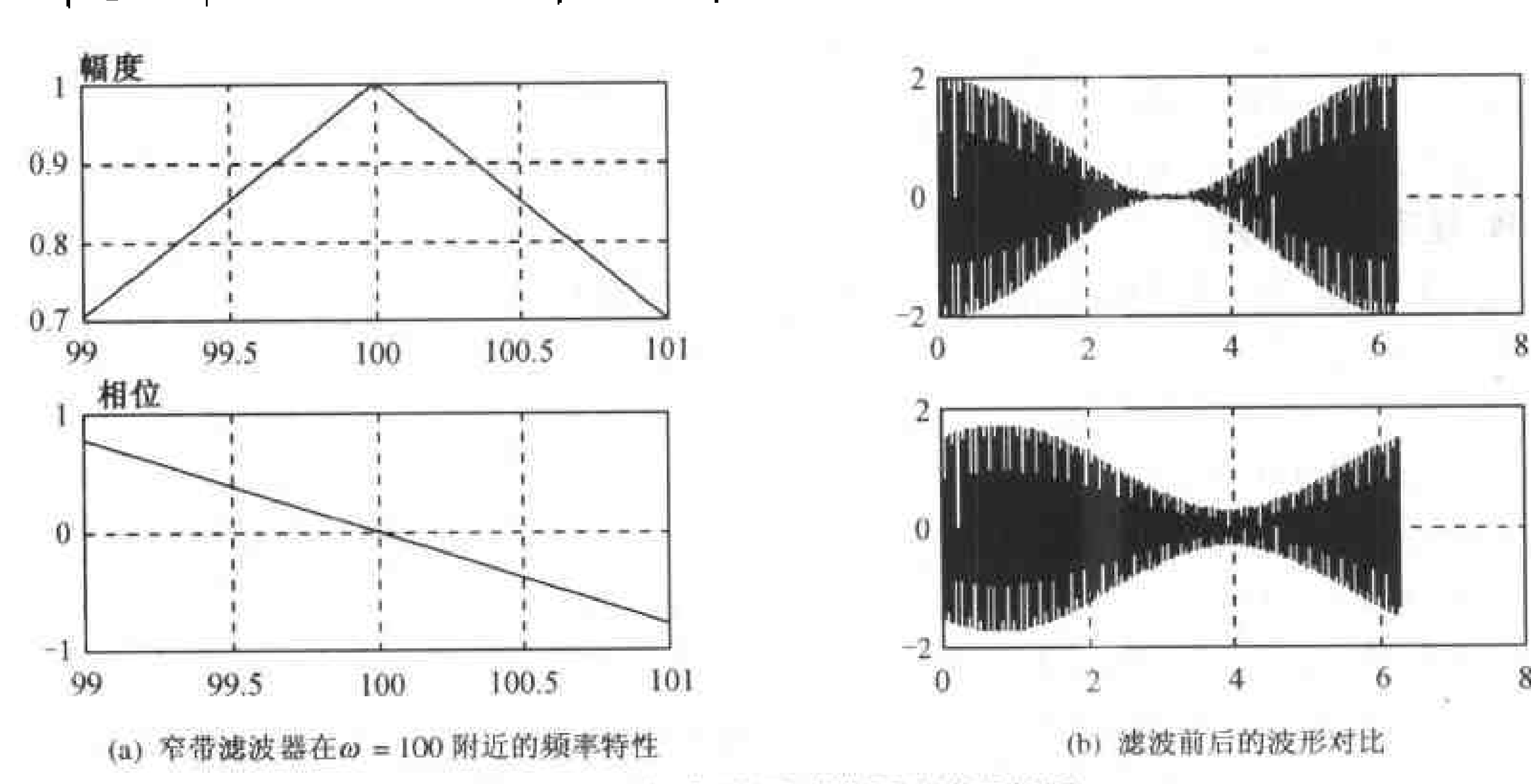
幅频和数频的响应分别为 abs $H(\mathrm{j}\omega)$，angle $H(\mathrm{j}\omega)$，如图 6.10 所示。

输出的相量为

$$\dot{U}_2(\mathrm{j}\omega)=\dot{U}_1(\mathrm{j}\omega)\cdot H(\mathrm{j}\omega)=\left|\dot{U}_2(\omega)\right|\mathrm{e}^{\mathrm{j}\varphi(\omega)}$$

稳态响应

$$u_2(t)=\left|U_2(99)\right|\cos(99t+\varphi(99))+\left|U_2(100)\right|\cos(100t+\varphi(100))+\left|U_2(101)\right|\cos(101t+\varphi(101))$$



(a) 窄带滤波器在 $\omega=100$ 附近的频率特性　　(b) 滤波前后的波形对比

图 6.10　调幅信号通过带通滤波器的图

■ MATLAB程序q610.m

```
clear
t=linspace(0, 2*pi, 1001);         % 信号周期为 2*pi，分成 1000 份
w=[99, 100, 101];                  % 输入信号的三个频率分量
U=[0.5, 1, 0.5];                   % 三个频率分量对应的向量（虚部为零）
b=[2, 0];a=[1, 2, 10001];          % 滤波器分子分母系数向量
u1=U*cos(w'*t+angle(U')*ones(1, 1001));   % 输入信号的时间曲线
H=polyval(b, j*w)./polyval(a, j*w);        % 求滤波器在三个频点上的频率响应，也可用
H=freqs(b, a, w);
% 画出滤波器的频率响应曲线，只用三个频点，图形不好看。
figure(1)
```

```
subplot(2, 1, 1), plot(w, abs(H)), grid              % 幅度
subplot(2, 1, 2), plot(w, angle(H)), grid        % 相位
u21=abs(U(1)*H(1))*cos(99*t+angle(U(1)*H(1)));  % 角频率为 99 的分量
u22=abs(U(2)*H(2))*cos(100*t+angle(U(2)*H(2))); % 角频率为 100 的分量
u23=abs(U(3)*H(3))*cos(101*t+angle(U(3)*H(3))); % 角频率为 101 的分量
u2=u21+u22+u23;                                     % 求和
% 巧妙地利用元素群运算和矩阵运算相结合的方法可把这四条语句合成一条如下
% u2=abs(U. *H) *cos(w'*t+angle((U.*H).')*ones(1, 1001))
% 注意对复数矩阵(U.*H), (U.*H)'为其共轭转置, (U.*H).'为转置而不共轭
figure(2)                             % 画出原信号和滤波后信号的波形作比较
subplot(2, 1, 1), plot(t, u1)
subplot(2, 1, 2), plot(t, u2)
```

■ **程序运行结果**

结果如图6.10(b)所示。

【例 6.11】 非周期信号(方波)的频谱分析

如图 6.11-1 的矩形脉冲信号，求其在$\omega=-40$rad/s～40rad/s 区间的频谱。

解：■ 建模

傅立叶变换表示式为

$$F(\mathrm{j}\omega)=\int_{-\infty}^{\infty}f(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \tag{6.5}$$

按 MATLAB 作数值计算的要求，它不能计算无限区间，根据信号波形的情况，将积分上下限定为 0～10s，并将 t 分成 N 等份，用求和代替积分。这样，上式可写为

$$\begin{aligned}F(\mathrm{j}\omega)&=\sum_{i=1}^{N}f(t_i)\mathrm{e}^{-\mathrm{j}\omega t_i}\Delta t\\&=[f(t_1),f(t_2),\cdots,f(t_n)][\mathrm{e}^{-\mathrm{j}\omega t_1},\mathrm{e}^{-\mathrm{j}\omega t_2},\cdots\mathrm{e}^{-\mathrm{j}\omega t_n}]'\Delta t\end{aligned} \tag{6.6}$$

这说明求和的问题可以用$f(t)$行向量乘以 $\mathrm{e}^{-\mathrm{j}\omega t_n}$ 列向量来实现。式中 Δt 是 t 的增量，在程序中，用 dt 表示。

由于求一系列不同ω处的 F 值，都用同一公式，这就可以利用 MATLAB 中的元素群运算能力。将ω设为一个行数组，代入（6.6）式，则可写为（程序中ω用 w 表示）

$$\mathrm{F}=\mathrm{f}*\exp(-\mathrm{j}*\mathrm{t}'*\mathrm{w})*\mathrm{dt} \tag{6.7}$$

其中，F 是与 w 等长的行向量，t'是列向量，w 是行向量，t'*w 是一矩阵，其行数与 t 相同，列数与 w 相同。这样（6.7）式就完成了傅立叶变换，类似地也可得到傅立叶逆变换表示式为

$$\mathrm{f}=\mathrm{F}*\exp(\mathrm{j}*\mathrm{w}'*\mathrm{t})*\mathrm{dw/pi}$$

即等价于

$$f(t)=\frac{1}{\pi}\int_{0}^{\infty}F(\mathrm{j}\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega$$

■ **MATLAB程序q611.m**

```
% 演示  (1)频域样本点数可取得与时域样本点数不同
% 演示  (2)若要求的频谱太宽, 而时域样本点数又取得太少, 会发生频率泄漏
clear, tf=10;
N = input('取时间分隔的点数 N= ');
```

```
dt = 10/N; t = [1:N]*dt;          % 给出时间分割
f =[ones(1, N/2), zeros(1, N/2)];      % 给出信号（此处是方波）
wf = input('需求的频谱宽度 wf= ');
Nf = input('需求的频谱点数 Nf= ');
w1 =linspace(0, wf, Nf);dw=wf/(Nf-1);
F1 = f*exp(-j*t'*w1) *dt;          % 求傅立叶变换
w = [-fliplr(w1), w1(2:Nf)];          % 补上负频率
F = [fliplr(F1), F1(2:Nf)];       % 补上负频率区的频谱
subplot(1, 2, 1), plot(t, f, 'linewidth', 1.5), grid
subplot(1, 2, 2), plot(w, abs(F), 'linewidth', 1.5), grid
```

■ **程序运行结果**

取时间分隔的点数 N = 256，需求的频谱宽度 wf = 40，需求的频谱点数 Nf = 64，得出图 6.11-1。

若取时间分隔的点数 N = 64，频谱宽度 wf = 40，频谱点数 Nf = 256，则得图 6.11-2。因此时采样周期为 dt = 10/64，对应的采样频率为 6.4Hz 或ω_s= 40rad/s。从图中可看出高频频谱以$\omega_s/2$ 处为基准线的转迭，称为频率泄漏。

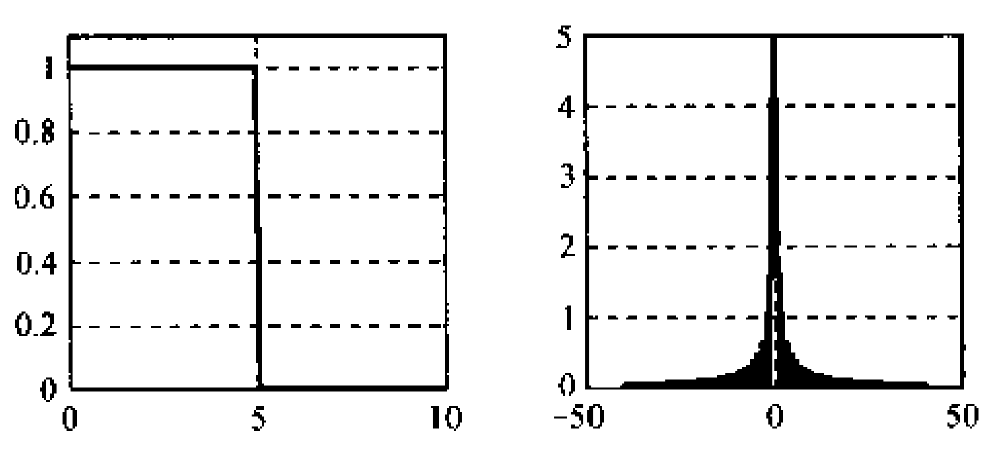


图 6.11-1　时域信号及其频谱图(采样密)

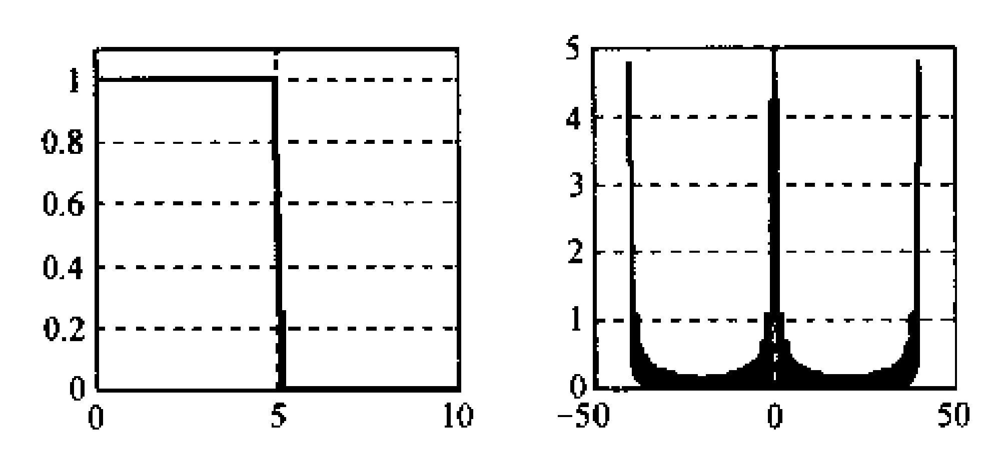


图 6.11-2　时域信号及其频谱图(采样稀，有频率泄漏)

【例 6.12】 用傅立叶变换计算滤波器的响应

计算幅度为 1，宽度为 5s 的矩形脉冲（同例 6.11）通过下列滤波器的响应。

（1）理想低通滤波器，其频率的响应

$$H(\mathrm{j}\omega)=\begin{cases}1 & -10<\omega<10\\ 0 & \omega<-10,\ \omega>10\end{cases}$$

（2）三阶巴特沃斯低通滤波器（截止角频率ω_c=10rad/s）转移函数

$$H(s)=\frac{500}{s^3+20s^2+200s+1000}$$

解：■ 建模

滤波器输出的频谱

$$Y(j\omega)=F(j\omega)\cdot H(j\omega)$$

其时间响应 $y(t)$ 是 $Y(j\omega)$ 的傅立叶反变换。

（1）理想低通滤波器的截止角频率 $\omega_c=10$，故只取 $F(j\omega)$ 中 $\omega=0\sim10$ 的部分，用 MATLAB 语言表述，在这个区域内的频率分量对应的 ω 的下标数组为

n2 = find ((w>= wc) & (w<= wc)

其对应的频率数组为

w2 = w(n2)

频段内的频谱数组为 F2 = F(n2:)，它就是滤波后的频谱数组 Y2，其逆变换即

y2 = F2*exp(j*n2'*t)/pi*dw

（2）三阶低通滤波器的频率响应

$$H(j\omega)=H(s)\Big|_{s=j\omega}=\frac{500}{(j\omega)^3+20(j\omega)^2+200(j\omega)+1000}$$
$$=\frac{500}{(1000-20\omega^2)+j(200\omega-\omega^3)}$$

滤波器的输出为

$$y_2(t)=F^{-1}[H(j\omega)F(j\omega)]$$

■ MATLAB程序q612a.m和q612b.m

```
% (1)q612a
%滤波器 1: 理想低通滤波器
q611g          % 调用例 6.11 程序, 设定数据为 N=256, wf=20, Nf=128
w2f=input('理想低通滤波器的带宽（1/秒）= ');
n2 = find((w>=w2f)&(w<w2f)); % 求出低通滤波器带宽内的下标
w2 = w(n2);                    % 取出中段频率
Y2 = F(n2);                  % 取出中段频谱
figure(1), subplot(1, 2, 1),
plot(w2, abs(Y2), 'linewidth', 1.5), grid % 画出滤波后的频谱
y2=Y2*exp(j*w2'*t)/pi*dw;               % 对中段频谱求傅立叶逆变换
subplot(1, 2, 2),
plot(t, f, t, y2, 'linewidth', 1.5), grid % 画出原波形及滤波后的波形
% (2)q612b
%滤波器 2: 截止频率为 wc=10 的三阶巴特沃斯低通滤波器转移函数为
q611g          % 调用例 6.11 程序, 设定数据为 N=256, wf=20, Nf=128
H = freqs(500, [1, 20, 200, 1000], w);
% 如用信号处理工具箱, 语句为
% [b, a]==butter(wn, 阶数); H=freqs(b, a, w),
```

```
Y3 = H.*F;
figure(2), subplot(1, 2, 1),
plot(w, abs(Y3), 'linewidth', 1.5), grid  % 画出滤波后的频谱
y3 = Y3*exp(j*w'*t)/pi*dw;              % 对中段频谱求傅立叶逆变换
subplot(1, 2, 2)
plot(t, f, t, y3, 'linewidth', 1.5), grid % 画出原波形及滤波后的波形
```

■ 程序运行结果

运行 q612a.m，设定理想滤波器带宽为 10 时所得结果如图 6.12-1。

运行 q612b.m，通过三阶巴特沃斯低通滤波器后所得结果如图 6.12-2。

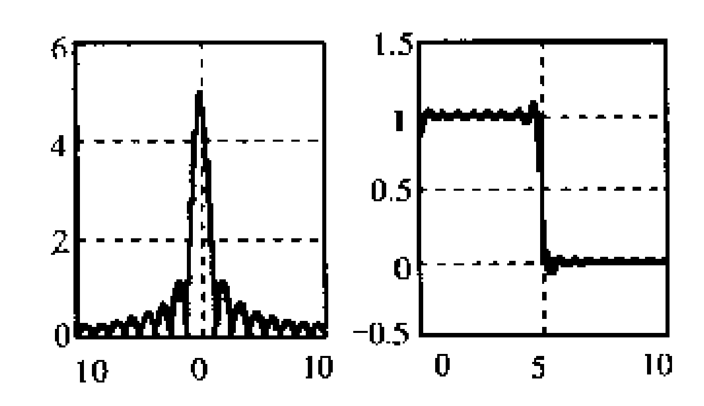


图 6.12-1　理想滤波后频谱和波形

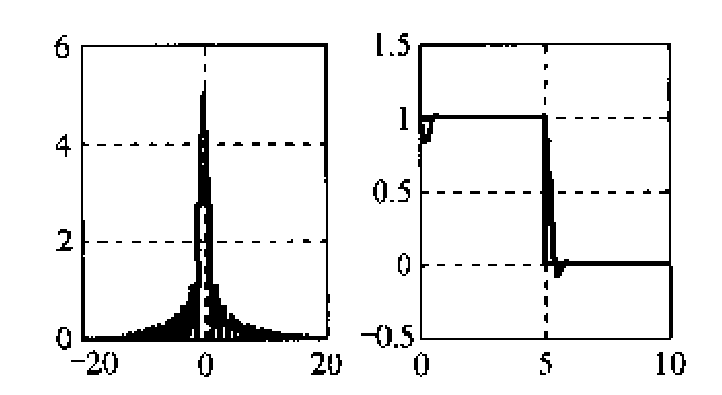


图 6.12-2　巴特沃斯滤波后的频谱和波形

6.3　离散信号和系统

本节讨论用 MATLAB 表示离散信号（序列）和线性时不变（LTI）离散系统的问题。由于 MATLAB 数值计算的特点，用它来分析离散的信号与系统是很方便的。在 MATLAB 中，可以用一个列向量来表示一个有限长度的序列。然而这样一个向量并没有包含采样位置的信息。因此，完全地表示 $x(n)$要用 $\boldsymbol{x}$ 和 $\boldsymbol{n}$ 两个向量，例如序列

$x(n)$={2，1，-1，3，1，4，3，7}（下面的箭头为第 0 个采样点）。

　　　　↑

在 MATLAB 中表示为：

n = [-3，-2，-1，0，1，2，3，4]；x=[2，1，-1，3，1，4，3，7]；

当不需要采样位置信息或这个信息是多余的时候（例如该序列从 n=0 开始），可以只用 $\boldsymbol{x}$ 向量来表示。由于内存有限，MATLAB 无法表示无限序列。

【例 6.13】 离散信号的 MATLAB 表述

编写 MATLAB 程序来产生下列基本脉冲序列：

（1）单位脉冲序列，起点 n_0，终点 n_f，在 n_s 处有一单位脉冲($n_0 \leqslant n_s \leqslant n_f$)。

（2）单位阶跃序列，起点 n_0，终点 n_f，在 n_s 前为 0，在 n_s 后为 1($n_0 \leqslant n_s \leqslant n_f$)。

（3）复数指数序列。

解：■ 建模

这些基本序列的表达式比较简明，编写程序也不难。对单位脉冲序列，此处提供了两种方法，其中用逻辑关系的编法比较简洁。读者可从中看到 MATLAB 编程的灵活性和技巧性，绘制脉冲序列，通常用 stem 语句。本例取 n=0，1，…，10

（1）单位脉冲序列（也叫单位序列、单位样值序列）

$$\delta(n)=\begin{cases}1 & n=0\\0 & 其余\end{cases}$$

迟延 n_s 的单位脉冲序列

$$x_1(n)=\delta(n-n_s)=\begin{cases}1 & n=n_s\\0 & 其余\end{cases}$$

本例取 $n_s=3$。

（2）单位阶跃序列

$$u(n)=\begin{cases}1 & n\geqslant 0\\0 & n<0\end{cases}$$

迟延 n_s 的单位阶跃序列

$$x_2(n)=u(n-n_s)=\begin{cases}1 & n\geqslant n_s\\0 & n<n_s\end{cases}$$

本例取 $n_s=3$。

（3）复指数序列

$$x_3(n)=\begin{cases}e^{(\alpha+\omega_w)n} & n\geqslant 0\\0 & n<0\end{cases}$$

当$\omega=0$时，它是实指数序列；当$\alpha=0$时，它是虚指数序列，其实部为余弦序列，虚部为正弦序列。本例取$\alpha=-0.2$，$\omega=0.5$。

■ **MATLAB程序q613.m**

```
clear, n0=0; nf=10; ns=3;
n1=n0:nf; x1=[zeros(1, ns-n0), 1, zeros(1, nf-ns)]; %单位脉冲序列的产生
% 用逻辑式产生单位脉冲序列更为简洁 n1 = n0:nf; x1=[(n1-ns)==0]
n2=n0:nf;x2=[zeros(1, ns-n0), ones(1, nf-ns+1)]; %单位阶跃序列的产生
% 用逻辑式产生单位阶跃序列的语句 n1 = n0:nf; x1=[(n1-ns)>=0]
n3 = n0:nf; x3=exp((-0.2+0.5j)*n3);   % 复数指数序列
subplot(2, 2, 1), stem(n1, x1);title('单位脉冲序列')
subplot(2, 2, 3), stem(n2, x2);title('单位阶跃序列')
subplot(2, 2, 2), stem(n3, real(x3));line([0, 10], [0, 0])
title('复指数序列'), ylabel('实部')
subplot(2, 2, 4), stem(n3, imag(x3));line([0, 10], [0, 0]),  %画横坐标
ylabel('虚部')
```

■ **程序运行结果**

见图 6.13。读者可与例 6.1 连续系统的结果图 6.1 相比较，观察两者在处理方法和图形表示上的不同。

【例 6.14】 差分方程的通用递推程序

描述线性时不变离散系统的差分方程为

$$a_1y(n)+a_2y(n-1)+\cdots+a_{na}y(n-n_a+1)=b_1u(n)+b_2u(n-1)+\cdots+b_{nb}u(n-n_b+1) \quad (6.8)$$

编写解上述方程的通用程序。

解：■ 建模

MATLAB 用递推法解差分方程是很方便的，因而通常不再用经典法或 z 变换法。

将方程（6.8）左端都移到等号右端可得（用 MATLAB 程序语言表示）

a(1)*y(n)=b(1)*u(n)+⋯+b(nb)*u(n−nb+1)−a(2)*y(n−1)−⋯−a(na)*y(n−na+1)

令 us=[u(n)，⋯，u(n−nb+1)]；　ys=[y(n−1)，⋯，y(n−na+1)]；

则上式可写为

a(1)*y(n) = b*us' − a(2:na)*ys'

于是得 y(n)=(b*us' − a(2::na)*ys')/a(1)

这个方程是可以递推计算的。

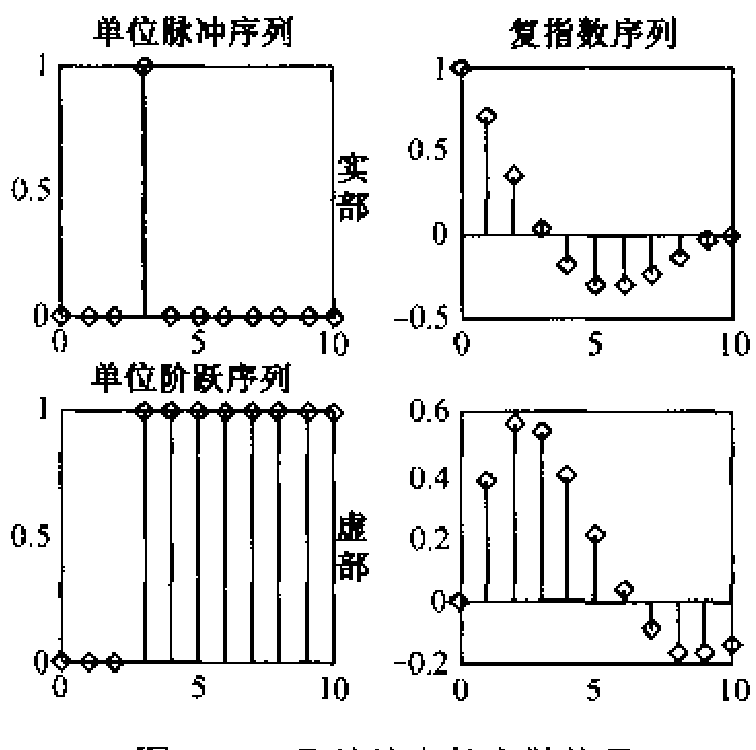




图 6.13　几种基本的离散信号

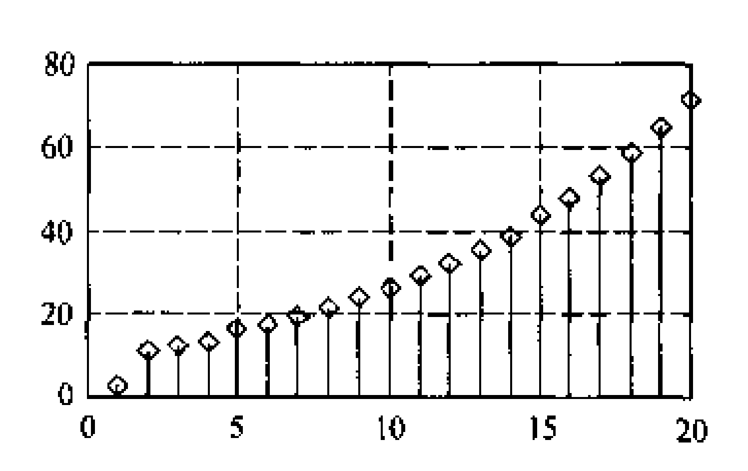


图 6.14　直接解差分方程得出的结果

这里遇到了一个困难，MATLAB 的变量下标不允许取负数，而这个公式需要知道 n = 0 之前的 y 和 u，本例中的处理方法是另设两个变量 ym 和 um，使 ym(k)=ys(k−na+1)，um(k) = us(k−na+1)，这相当于把 y 和 u 右移 na−1 个序号，故 ym 和 um 的第 1 到 na−1 位相当于 y 和 u 在起点之前的初值。注意在程序中，随着计算点的右移，要随时更新相应于公式中的向量 us 和 ys。

■ MATLAB程序q614.m

```
a = input('差分方程左端的系数向量 a=[a(1), ...a(na)]= ');
b = input('差分方程右端的系数向量 b=[b(1), ...b(na)]= ');
u = input('输入信号序列 u= ');
na=length(a);nb=length(b);nu=length(u);
s=['起算点前', int2str(na-1), '点 y 的值 =[y(', int2str(na-2), '),... ,y(0)]= '];
ym=zeros(1, na+nu-1);ym(1:na-1) = input(s);  % 建立 ym 序列并赋予初值
um = [zeros(1, na-2), u];                    % 建立 um 序列，设 u 以 n=0 为起点输入
for n=na:na+nu-1                             % 这个 n 以 ym 的起点为准
   ys = ym(n-1:-1:n-na+1); us = um(n:-1:n-nb+1);    % 生成 us 及 ys
   ym(n) = (b*us'-a(2:na)*ys')/a(1);         % 差分方程递推求 ym
end
% 把 ym 时间坐标右移 na-1 位，求出 y
y = ym(na:na+nu-1); stem(y), grid on  % 画出波形
line([0,nu], [0,0])           % 画横坐标轴
```

■ **程序运行结果**

执行此程序，输入

```
a=[1, 0.1, 0.15, -0.225];b=[3, 7, 1]
u=exp(0.1*[1:20]);及 ym=[0, 0, 0];
```

得出的结果如图 6.14 所示。

MATLAB 函数 filter 与此程序等效。

键入

```
y1=filter(b, a, u); stem(y1)
```

可以得到同样的结果，因此以后可直接用此函数。

【例 6.15】 求离散系统在各种输入下的响应

描述 LTI 系统的差分方程为

$$y(n)-y(n-1)+0.9y(n-2)-0.5y(n-3)=5u(n)-2u(n-1)+2u(n-2)$$

（1）如已知 $y(0)=-2$，$y(-1)=2$，$y(-2)=-\frac{1}{2}$，求零输入的响应，计算 20 步。

（2）求单位脉冲的响应 $h(n)$，计算 20 步。

（3）求单位阶跃的响应 $g(n)$，计算 20 步。

解：■ 建模（利用例 6.14 的通用程序）

（1）求零输入响应，即输入为零时，仅由初始状态产生的响应。

令 $\boldsymbol{a}=[1, -1, 0.9, -0.5]$; $\boldsymbol{b}=[5, -2, 2]$;

则 us = zeros(1, 20); （输入信号长度决定要求的输出长度，此处取 20）

ym = [−1/2, 2, −2]; （ym 长度应为系数 a 的长度减 1）

（2）单位脉冲响应 $h(n)$是输入为$\delta(n)$时的零状态响应。

令 $y(0)=y(-1)=y(-2)=0$，则 ym = [0, 0, 0]

$u(n)=\delta(n)$，则 us = [1, zeros(1, 19)]

（3）阶跃响应是输入为单位阶跃序列$\varepsilon(n)$时的零状态响应。

令 $y(0)=y(-1)=y(-2)=0$，$u(n)=\varepsilon(n)$

则 ym = [0, 0, 0]; um = ones (1, 20)

在信号处理工具箱中 filter 函数与此程序有同样功能。

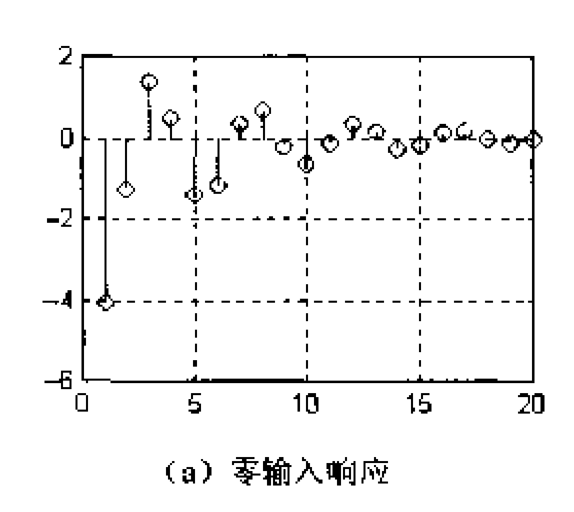
■ **MATLAB程序q615.m**

```
% 题 614 已给出的求差分方程解的普遍程序，差别仅是输入序列的不同
% 初始条件反映在 ym 上，输入反映在 u 上，u 的长度即计算序列的长度
disp('分题（1），初始条件响应')
figure(1), q614 ;
% 按提示键入 a=[1, -1, 0.9, -0.5]; b=[5,-2,2];
% u=zeros(1, 20); ym=[-1/2,2,-2];
disp('分题（2），单位脉冲响应');
figure(2), q614;
% 键入 a, b 同上, 及 ym=zeros(1, 3);u=[1, zeros(1, 19)];
disp('分题（3），单位阶跃响应');
figure(3), q614;
```

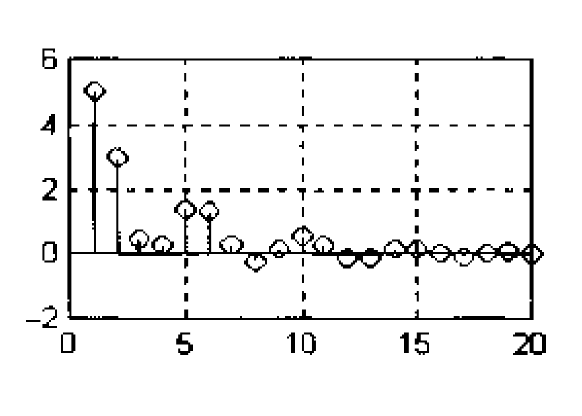
```
% 键入 a, b 同上, 及 ym=zeros(1, 3); u=[ones(1, 20)];
```

■ **程序运行结果**

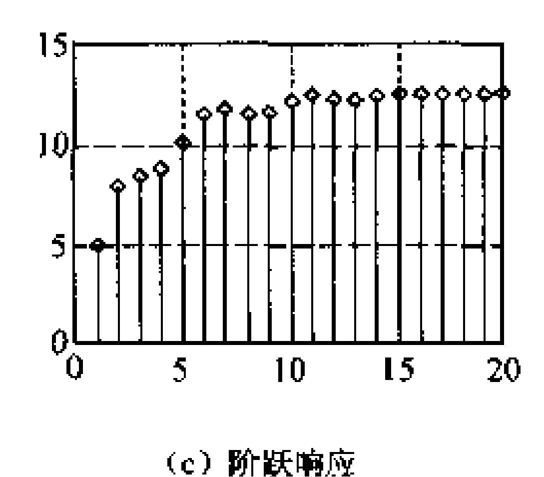
见图 6.15。



(a) 零输入响应



(b) 脉冲响应



(c) 阶跃响应

图 6.15 离散系统的时域解

【例 6.16】 二阶巴特沃斯低通数字滤波器的频率响应。

二阶巴特沃斯低通滤波器的系统函数（传递函数）为

$$H(z)=\frac{z^2+2z+1}{(2+\sqrt{2})z^2+(2-\sqrt{2})}$$

求其频率响应并做图（0～2π）。

解：■ 建模

离散系统的频率响应函数为（这里$\theta=\omega t_s$，t_s为抽样周期）

$$H(\mathrm{e}^{\mathrm{j}\theta})=H(z)\big|_{z=\mathrm{e}^{\mathrm{j}\theta}}$$

其幅频特性为$\left|H(\mathrm{e}^{\mathrm{j}\theta})\right|$，相频特性为$\angle H(\mathrm{e}^{\mathrm{j}\theta})$，下面按定义编程。信号处理工具箱中的 freqz 函数与此程序有同样功能。

■ **MATLAB程序q616.m**

```
% 二阶巴特沃斯低通滤波器的离散系统函数为:
% H(z)=(z^2+2z+1)/((2+sqrt(2))z^2+(2-sqrt(2));
% 求其频率响应可将 z 用 exp(i*w) 代入
b=[1, 2, 1]; a=[2+sqrt(2), 0, 2-sqrt(2)]; % 给出滤波器分子分母系数
N=input('取频率数组的点数 N= ');
w=[0:N-1]*pi/N;                              % 给出 0 到 pi 之间的频率数组
H=polyval(b, exp(i*w))./polyval(a, exp(i*w));  % 求频率响应
figure(1)                          % 在线性坐标内画频率特性
subplot(2, 1, 1), plot(w, abs(H)), grid
subplot(2, 1, 2), plot(w, unwrap(angle(H))), grid
figure(2)                          % 在对数坐标内画频率特性
subplot(2, 1, 1), semilogx(w, 20*log10(abs(H))), grid
subplot(2, 1, 2), semilogx(w, unwrap(angle(H))), grid
```

■ **程序运行结果**

按提示输入点数（例如 100），可得到图 6.16 的频率特性曲线。

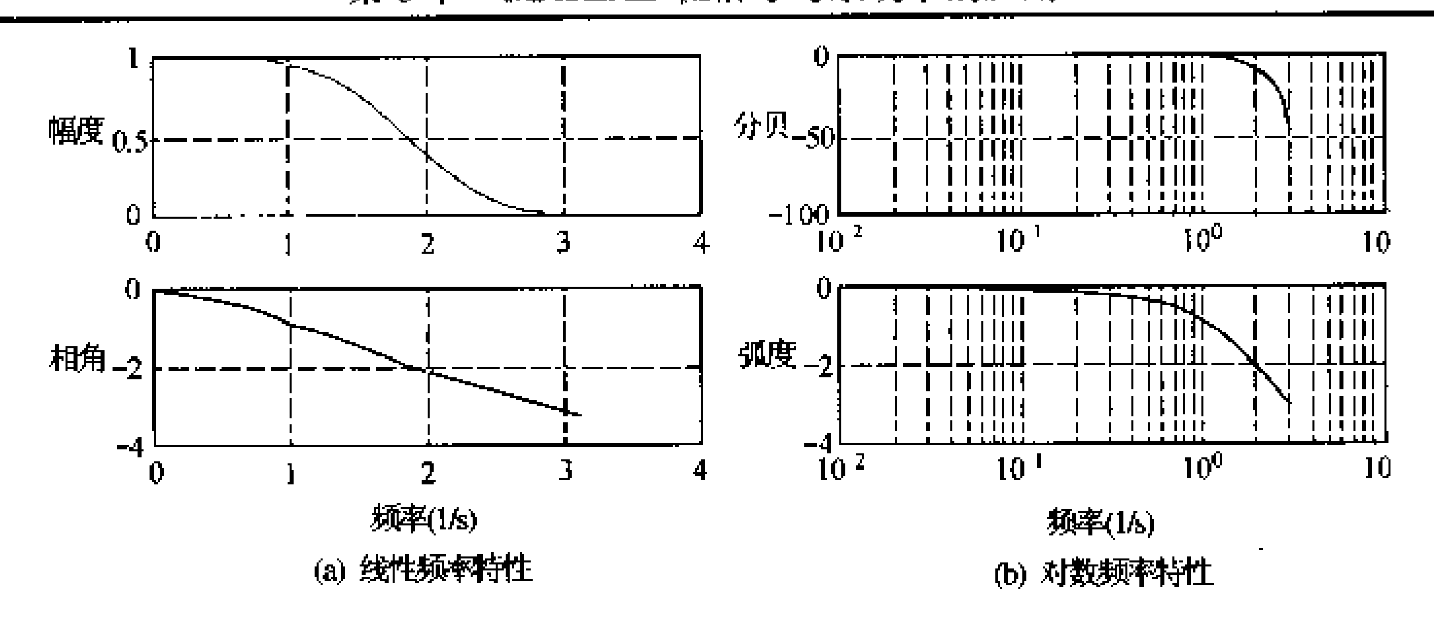


图 6.16　二阶巴特沃斯低通数字滤波器的频率响应

6.4　线性时不变系统的模型

6.4.1　模型的典型表达式

LTI 系统模型的典型表达式有状态空间型、传递函数型、零极点增益型等，它们都能描述系统的特性，但各有不同的应用场合。熟悉各表示式的互相转换是非常重要的。

1．连续系统

● 状态空间型

设 x 为状态变量，u 为输入，y 为输出，系统的状态方程为(为免混淆，此处分子、分母多项式用 f 和 g)

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

如果系统是 n 阶的，输入有 n_u 个，输出有 n_y 个，则 $\boldsymbol{A}$ 为 $n\times n$ 阶，$\boldsymbol{B}$ 为 $n\times n_u$ 阶，$\boldsymbol{C}$ 为 $n_y\times n$ 阶，而 $\boldsymbol{D}$ 为 $n_y\times n_u$ 阶矩阵，对单输入单输出(SISO)系统，$n_y=n_u=1$。已知 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$ 四个矩阵，即可建立系统模型。

● 传递函数型

单输入单输出 n 阶系统的传递函数为

$$H(s)=\frac{f(1)s^m+f(2)s^{m-1}+\cdots+f(m)s+f(m+1)}{g(1)s^n+g(2)s^{n-1}+\cdots+g(n)s+g(n+1)}=\frac{f(s)}{g(s)}$$

对于 SISO 系统，$f(s)$是 s 的 m 次多项式，$g(s)$是 s 的 n 次多项式。对于多输入单输出(MISO)系统，设有 n_u 个输入，$H(s)$将是由 n_u 个多项式分式构成的单列矩阵；对于多输入多输出(MIMO)系统，$H(s)$将是由 $n_y\times n_u$ 个多项式分式构成的多项式矩阵。在通常的 SISO 系统中，考虑到分子分母可以对 $g(1)$约分，使分母上的系数 $g(1)=1$，于是有

$$f(s)=f(1)s^m+f(2)s^{m-1}+\cdots+f(m)s+f(m+1) \qquad (6.9)$$

$$g(s)=s^n+g(2)s^{n-1}+\cdots+g(n)s+g(n+1) \qquad (6.10)$$

因此，知道分子系数矢量 $\boldsymbol{f}=[f(1),f(2),\cdots,f(m+1)]$ 和分母系数矢量 $\boldsymbol{g}=[g(1),g(2),\cdots,$

$g(n+1)$]，就惟一地确定了系统的模型（注意系统的阶次 n）。而对物理可实现的系统，必有 $n \geqslant m$。

● 零极增益型

对（6.9）及（6.10）进行因式分解，可得

$$H(s)=\frac{k(s-z(1))(s-z(2))\cdots(s-z(m))}{(s-p(1))(s-p(2))\cdots(s-p(n))} \tag{6.11}$$

令 $\boldsymbol{z}$ = [$z(1)$，$z(2)$，…，$z(m)$]为系统的零点矢量，$\boldsymbol{p}$ = [$p(1)$，$p(2)$，…，$p(n)$] 为系统的极点矢量，k 为系统增益，它是一个标量。可以看出，$H(s)$有 m 个零点、n 个极点。物理可实现系统的 $n \geqslant m$，系统的模型将由矢量 $\boldsymbol{z}$，$\boldsymbol{p}$ 及增益 k 惟一确定，故称为零极增益模型。零极增益模型通常用于描述 SISO 系统。并可以推广到 MISO 系统。

● 极点留数型

如果式（6.11）中的极点都是单极点，将零极增益模型分解为部分分式，可得

$$H(s)=\frac{r(1)}{s-p(1)}+\frac{r(2)}{s-p(2)}+\cdots+\frac{r(n)}{s-p(n)}+h \tag{6.12}$$

其中 $\boldsymbol{p}$ = [$p(1)$，$p(2)$，…，$p(n)$] 仍为极点矢量，而 $\boldsymbol{r}$ = [$r(1)$，$r(2)$，…，$r(n)$] 为对应于各极点的留数矢量，$\boldsymbol{p}$，$\boldsymbol{r}$ 两个矢量及常数 h 惟一地决定了系统的模型。

下面来比较一下这四种情况下模型系数的总个数。假定都是 SISO 系统，阶数为 n，则状态空间型有 n^2+2n+1 个系数；传递函数型为 $m+n+1$ 个(不含 $g(1)$) (注意，由于 $m \leqslant n$ 时系数的数目小于等于 $2n+1$)；零极增益型的系数个数为 $n+m+1$；而极点留数型为 $2n+1$。因此，传递函数法的待定系数最少，而状态空间法的待定系数最多。这说明了状态空间法中有许多冗余的系数。事实上，同一个系统可以有无数个状态空间矩阵 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$ 的组合来描述，其他描述方法则都是惟一的。下面在讨论这几种模型相互变换时将进一步论及这个问题。

2. 离散系统

以上四种表示模型的方法可以全部推广至离散系统。为了区别，将所有的系数矩阵后面加小写字母 d，便有

● 状态空间型

$$\boldsymbol{x}[n+1]=\boldsymbol{A}\mathrm{d}\boldsymbol{x}[n]+\boldsymbol{B}\mathrm{d}\boldsymbol{u}[n] \tag{6.13}$$

$$\boldsymbol{y}[n]=\boldsymbol{C}\mathrm{d}\boldsymbol{x}[n]+\boldsymbol{D}\mathrm{d}\boldsymbol{u}[n] \tag{6.14}$$

● 传递函数型

$$H(z)=\frac{\boldsymbol{f}\mathrm{d}(1)z^{m}+\boldsymbol{f}\mathrm{d}(2)z^{m-1}+\cdots+\boldsymbol{f}\mathrm{d}(m)z+\boldsymbol{f}\mathrm{d}(m+1)}{z^{n}+\boldsymbol{g}\mathrm{d}(2)z^{n-1}+\cdots+\boldsymbol{g}\mathrm{d}(n)z+\boldsymbol{g}\mathrm{d}(n+1)} \tag{6.15}$$

● 零极增益型

$$H(z)=\frac{\boldsymbol{k}\mathrm{d}(z-\mathrm{zd}(1))(z-\mathrm{zd}(2))\cdots(z-\mathrm{zd}(m))}{(z-\boldsymbol{p}\mathrm{d}(1))(z-\boldsymbol{p}\mathrm{d}(2))\cdots(z-\boldsymbol{p}\mathrm{d}(n))} \tag{6.16}$$

● 极点留数型

$$H(z)=\frac{\boldsymbol{r}\mathrm{d}(1)}{z-\boldsymbol{p}\mathrm{d}(1)}+\frac{\boldsymbol{r}\mathrm{d}(2)}{z-\boldsymbol{p}\mathrm{d}(2)}+\cdots+\frac{\boldsymbol{r}\mathrm{d}(n)}{z-\boldsymbol{p}\mathrm{d}(n)}+\boldsymbol{h}\mathrm{d} \tag{6.17}$$

各种表述方法中参数阵的阶数和参数个数的讨论都与连续系统相同。

● 数字信号处理模型

在数字信号处理中，常常会用到按 s^{-1} 的降幂排列的传递函数，将（6.15）式的分子、

分母同除以 s^n，可得

$$H(s^{-1})=\frac{fd(1)s^{m-n}+fd(2)s^{m-n-1}+\cdots+fd(m)s^{1-n}+fd(m+1)s^{-n}}{1+gd(2)s^{-1}+\cdots+gd(n)s^{1-n}+gd(n+1)s^{-n}} \tag{6.18}$$

这可看做是传递函数法的一种变型。

● 二阶环节型

系统中经常会包含复数的零极点，这时用零极增益法表示就非常烦琐。对于实系数多项式，其复数的零极点必定是共轭的，把每一对共轭极点或零点多项式合并，就可得出多个二阶环节。

$$H(s^{-1})=\frac{(b_{01}+b_{11}s^{-1}+b_{21}s^{-2})\cdots(b_{0L}+b_{1L}s^{-1}+b_{2L}s^{-2})}{(a_{01}+a_{11}s^{-1}+a_{21}s^{-2})\cdots(a_{0L}+a_{1L}s^{-1}+a_{2L}s^{-2})} \tag{6.19}$$

这可看做是零极增益法的一种变型。

表 6.1 列出了连续和离散线性系统的各种模型表达式。

表 6.1　线性系统模型及其表述矩阵

表述方法	连续系统	表述矩阵	离散系统	表述矩阵
状态空间型	$\dot{x}=Ax+Bu$ $y=Cx+Du$	A，B， C，D	$x[n+1]=Ad\,x[n]+Bd\,u[n]$ $x[n]=Cd\,x[n]+Dd\,u[n]$	Ad，Bd Cd，Dd
传递函数型	$\frac{f(1)s^m+\cdots+f(m)s+f(m+1)}{s^n+\cdots+g(n)s+g(n+1)}$	f，g	$\frac{fd(1)z^m+\cdots+fd(m+1)}{z^n+\cdots+gd(n+1)}$	fd，gd
零极增益型	$\frac{k(s-z(1))(s-z(2))\cdots(s-z(m))}{(s-p(1))(s-p(2))\cdots(s-p(n))}$	z，p， k	$\frac{kd(z-zd(1))\cdots(z-zd(m))}{(z-pd(1))\cdots(z-pd(n))}$	zd，pd Kd
极点留数型	$\frac{r(1)}{s-p(1)}+\cdots+\frac{r(n)}{s-p(n)}+h$	r，p， h	$\frac{rd(1)}{z-pd(1)}+\cdots+\frac{rd(n)}{z-pd(n)}+hd$	rd，pd hd

6.4.2　模型转换

在连续系统的四种表达式之间可以任意进行转换。也就是说，知道其中任意的一组系数表述矩阵，就可以求出该系统在另一种表述法中的系数矩阵。这种变换如果用手算来进行，是非常繁杂和费时的，并且极易出错，因而计算机辅助是十分有益的。

可以把零极增益型和极点留数型都当成一类，即模域的表示法。这里主要研究时域、频域和模域这三者之间的变换，如图 6.17 所示。这种转换的运算在 MATLAB 语言中的函数名称也在图中标出。名称中出现的“2”字，应按英文谐音读作“to”，则 ss2tf 就表示由状态空间型转换为传递函数型，这样，在三个域之间的转换函数共有六个。再加上模域中的两种表述方法之间的转换函数 residue，通常有八种转换函数。此外，状态空间的表述不是惟一的，它可以有许多种形式，例如可控标准型、可观标准型、约当标准型等，可以用 ss2ss 来转换。图 6.17 中表示了这八种转换函数，其中 residue 具有双向变换功能。

这些函数的输入变量是变换源的参数阵，输出变量则是变换后的参数阵。例如

➢ [f，g] =ss2tf(A，B，C，D)

就实现了状态空间表示的模型矩阵转变为传递函数的分子、分母系数矢量。此处对这些变换的原理略作一点说明。

● 传递函数型到零极增益型

已知 f，g，求 z，p，k，即知道多项式求根。可用 MATLAB 内部函数 roots，即有

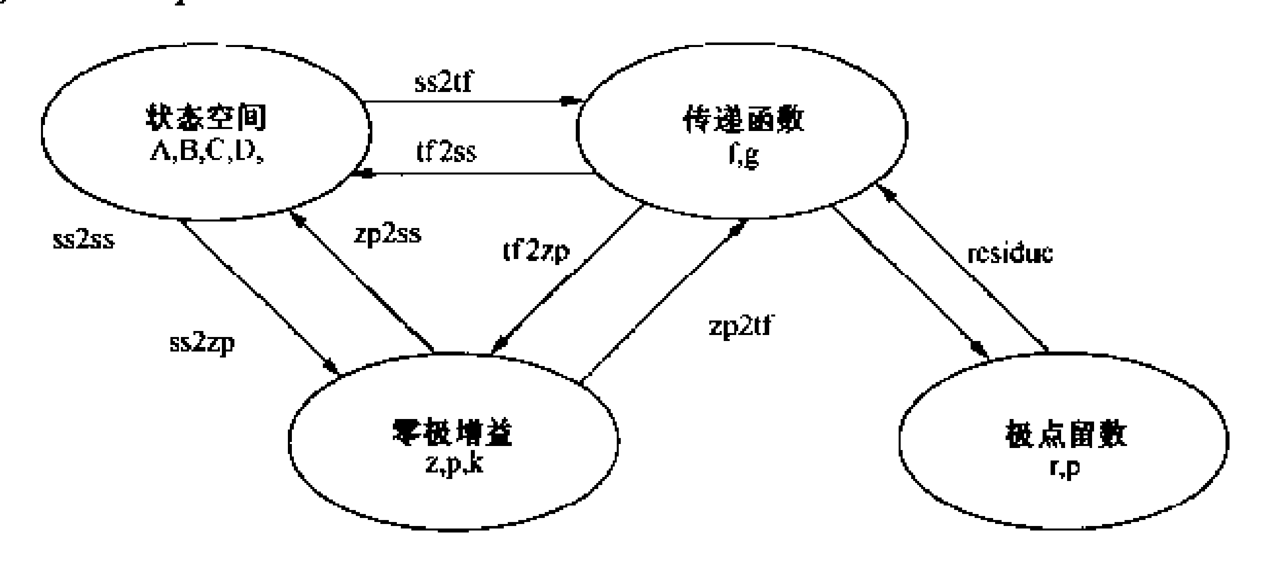


图 6.17　模型转换的八种函数

➢ z=roots(f)，p=roots(g)，k= f(1)/g(1)

这样就完成了[z，p，k] = tf2zp(f，g)的运算。

● 零极增益型到传递函数型

已知 z，p，k，求 f，g，即已知根求多项式。可用 MATLAB 内部函数 poly，它是 roots 的逆运算，即有

➢ f=poly(z)*k，g=poly(p)

这样就完成了[f，g] = zp2tf(z，p，k)的运算。

● 传递函数型到极点留数型

知道传递函数的系数 g 求其极点 p，方法同上，而求其中某极点处留数的公式为（对单极点而言）：

$$r_i = H(s)(s-p_i)\Big|_{s=p_i} = \frac{f(p_i)}{(p_i-p_1)(p_i-p_2)\cdots(p_i-p_n)}$$

其中　$f(p) = f_1p^{n-1} + f_2p^{n-2} + \cdots + f_{n-1}p + f_n$

MATLAB 已把这个复杂的运算过程编成专用函数 residue，格式为

➢ [r，p，h]=residue(f，g)

可直接由 f，g 求出 r，p，k。由极点留数型到传递函数型仍可用同一函数。

➢ [f，g] = residue(r，p，h)

residue 函数根据输入变元的数目为二或三个，决定变换的方向。

从状态空间型到传递函数型及零极增益型之间的相互转换，推导起来要麻烦一些（本书把从状态空间型到传递函数型的推导放在例 6.18 中，有兴趣的读者可参阅该例）。一般读者可以直接承认和调用 ss2tf，tf2ss，ss2zp，zp2ss 等函数（这些函数都在多项式函数库中），这些函数也适用于各种离散线性模型之间的变换。

【例 6.17】 由传递函数模型转换为零极增益和状态空间模型

已知描述系统的微分方程为

（1）$2\dddot{y}+3\ddot{y}+5\dot{y}+9y=2\ddot{u}-5\dot{u}+3u$

（2）$\dddot{y}+5\ddot{y}+7\dot{y}+3y=\ddot{u}+3\dot{u}+2u$

求出它的传递函数模型、零极增益模型、极点留数模型和状态空间模型。

解：■ 建模

本题的传递函数模型是显而易见的。

（1）f＝[2，−5，3]; g＝[2，3，5，9];

即
$$H(s)=\frac{2s^2-5s+3}{2s^3+3s^2+5s+9}$$

（2）f＝[1，3，2]; g＝[1，5，7，3];

其他模型均可用 tf2zp，tf2ss 及 residue 函数求得。为了检验本节中提供的变换程序，这里也列出了自编程序的解答以作比较。

■ **MATLAB 程序 q617.m**

```
% 由传递函数模型转为其他模型
format compact
f=input('传递函数分子系数数组 f= [f(1), ..., f(m+1)] =  ');
g=input('传递函数分母系数数组 g= [g(1), g(2), ..., g(n+1)] =  ');
printsys(f, g, 's')
disp('转为零极增益模型: ')
z=roots(f)
p=roots(g)
k=f(min(find(f(:)~=0)))/g(1)
[z1, p1, k1]=tf2zp(f, g)
disp('转为零极留数模型: ')
[r, p, h]=residue(f, g)
disp('转为状态空间模型: ')
[A, B, C, D]=tf2ss(f, g)
printsys(A, B, C, D)
```

■ **程序运行结果**

输入第（1）组 f，g 参数，得

```
传递函数分子系数数组
f =[f(1), ..., f(m)] =   [2, -5, 3]
传递函数分母系数数组
g =[g(1), g(2), ..., g(n)] =   [2, 3, 5, 9]
传递函数
num/den =
        2 s^2 - 5 s + 3
   ----------------------
   2 s^3 + 3 s^2 + 5 s + 9
转为零极增益模型
z = 1.5000
    1.0000
p = -1.6441
     0.0721 + 1.6528i
     0.0721 - 1.6528i
```

```
k = 1
z1 =
    1.5000
    1.0000
p1 =
   -1.6441
    0.0721 + 1.6528i
    0.0721 - 1.6528i
k1 =
     1
```

转为零极留数模型

```
r =  -0.2322 + 0.4716i
     -0.2322 - 0.4716i
      1.4644
p =    0.0721 + 1.6528i
       0.0721 - 1.6528i
      -1.6441
h =  [ ]
```

转为状态空间模型

```
 a =
            -1.50000        -2.50000      -4.50000
             1.00000         0             0
             0               1.00000       0
 b =
             1.00000
             0
             0
 c =
             1.00000     -2.50000     1.50000
 d =
             0
```

写成便于阅读的形式

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -1.5 & -2.5 & -4.5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u_t$$

$$y = [1 \quad -2.5 \quad 1.5] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

输入第（2）组 f，g 参数，得

零极增益模型

```
z =[ -2; -1],    p =[ -3.0000;   -1.0000 ;   -1.0000 ],    k = 1
```

其他模型不再列出。

注意本题参数（2）的零极点中都有-1，写成多项式时，分子分母中相同因式本应该消掉，但 MATLAB 作数值计算时，它看到的是根的值，而并非是 s 的因式，不可能具有因式相消的能力，因此需要人的参与、帮助、推理。本系统的零极增益模型的正确结果应为

```
z=-2,  p=[-3; -1], k=1
```

在符号运算（Symbolic）工具箱中，MATLAB 已增加了这种功能。

【例 6.18】 由状态空间型转换为传递函数型

设单输入单输出（SISO）线性时不变系统的状态空间表示式为

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

如果系统是 n 阶的，则 $\boldsymbol{A}$ 为 $n\times n$ 阶，$\boldsymbol{B}$ 为 $n\times 1$ 阶，$\boldsymbol{C}$ 为 $1\times n$ 阶，而 $\boldsymbol{D}$ 为 1×1 阶。给定 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$，就建立了系统模型。又设系统传递函数分子分母多项式的系数向量为 $\boldsymbol{f}$ 和 $\boldsymbol{g}$，现在的问题是已知 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$，如何求出 $\boldsymbol{f}$，$\boldsymbol{g}$？或反之，已知 $\boldsymbol{f}$，$\boldsymbol{g}$，如何求出 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$。由于状态空间表示法中有冗余参数，因此 $\boldsymbol{g}$，$\boldsymbol{f}$ 是惟一的；反之，则有无穷多解。

解：■ 建模

由 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$ 求传递函数分子、分母矢量 $\boldsymbol{f}$ 和 $\boldsymbol{g}$。

对状态方程取拉普拉斯变换，解出 $H(s)=f(s)/g(s)$，得

$$H(s)=\frac{f(s)}{g(s)}=\boldsymbol{C}(s\boldsymbol{I}+\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}=\frac{\boldsymbol{C}\mathrm{adj}(s\boldsymbol{I}-\boldsymbol{A})\boldsymbol{B}}{\det(s\boldsymbol{I}-\boldsymbol{A})}+\boldsymbol{D} \tag{6.20}$$

式中

$$(s\boldsymbol{I}-\boldsymbol{A})^{-1}=\frac{\mathrm{adj}(s\boldsymbol{I}-\boldsymbol{A})}{\det(s\boldsymbol{I}-\boldsymbol{A})} \tag{6.21}$$

利用等式(1)可以确定状态空间矩阵与传递函数系数之间的关系。令等式两端分母相等，有 $g(s)=\det(s\boldsymbol{I}-\boldsymbol{A})$，令全等式两端分子相等，有 $f(s)=\boldsymbol{C}\mathrm{adj}(s\boldsymbol{I}-\boldsymbol{A})\boldsymbol{B}+\boldsymbol{D}g(s)$。

传递函数的分母 $g(s)$ 可分解为其特征根 λ 的因式的乘积，即其多项式系数向量 g = poly(λ)，而特征根又可由 λ = eig(A) 求得。于是有

$$\mathrm{g = poly(eig(A)) = det(sI-A);} \tag{6.22}$$

由此式即可求出系数向量 $\boldsymbol{g}$，它的首项系数为 1。

令 $(s\boldsymbol{I}-\boldsymbol{A})$ 的伴随矩阵为 $\boldsymbol{P}(s)$，即

$$\boldsymbol{P}(\mathrm{s})=\mathrm{adj}(s\boldsymbol{I}-\boldsymbol{A})=\boldsymbol{P}_1 s^{n-1}+\boldsymbol{P}_2 s^{n-2}+\cdots+\boldsymbol{P}_{n-1}s+\boldsymbol{P}_n \tag{6.23}$$

它是按 s 的降幂排列的以 $n\times n$ 阶方阵 $\boldsymbol{P}_1$，$\boldsymbol{P}_2$，…，$\boldsymbol{P}_n$ 为系数的多项式，s 的最高幂次为 $(n-1)$，即它比 $\det(s\boldsymbol{I}-\boldsymbol{A})$ 低一阶。

将(6.21)式两端左乘以 $(s\boldsymbol{I}-\boldsymbol{A})$，得

$$\boldsymbol{I}=(s\boldsymbol{I}-\boldsymbol{A})\frac{\mathrm{adj}(s\boldsymbol{I}-\boldsymbol{A})}{\det(s\boldsymbol{I}-\boldsymbol{A})}=(s\boldsymbol{I}-\boldsymbol{A})\frac{\boldsymbol{P}(s)}{g(s)} \tag{6.24}$$

由此得

$$s\boldsymbol{P}(\mathrm{s})-\boldsymbol{A}\boldsymbol{P}(\mathrm{s})=g(s) \tag{6.25}$$

使等式两瑞 s 同次幂系数相等，可得以下的递推方程

s^n:　　$\boldsymbol{P}_1=\boldsymbol{I}$

s^{n-1}:　　$P_2 = AP_1 + g_2 I$

…　　………………

s^{n-k}:　　$P_{k+1} = AP_k + g_{k+1} I$

…　　………………

由这些递推方程确定 P，注意 P 本身是一个方阵，还要赋予下标，表示有 n 个系数方阵，这就需要三维的矩阵表示方法。求出 P 后，再由等式两端分子各对应项系数相等求出

f = C*P*B+D*g

在这个式子中，f 和 g 长度为 n+1，而 C*P*B 长度为 n，所以第一个等式为 f(1) = D*g(1)，以后的递推关系为 f(i+1)= C*P(:，:，i)* B+B*g(i+1)。

■ MATLAB程序q618.m

```
clear,
disp('输入状态方程系数矩阵 A, B, C, D')
A=input ('A= ')
B=input ('B= '),
C=input ('C=  '),
D=input ('D=  ')
g=poly(eig(A)); n=length(A); % 分母多项式系数由特征根求得
P(:, :, 1)=eye(n);
f(1)=D*g(1);                                % 分子多项式系数由递推求得
f(2)=C*P(:, :, 1 ) *B+D*g(2);
for i=2:n
   P(:, :, i )=A*P(:, :, i-1 )+g(i)*eye(n);
   f(i+1)=C*P(:, :, i ) *B+D*g(i+1);
end
f, g
```

在这个程序中使用了三维矩阵 P(:，:，i)，这种多维矩阵只能在 MATLAB 5.x 或更高的环境下才能运行。

■ **程序运行结果**

给定四阶系统的系数矩阵为：

```
A =0.2844    0.5828    0.4329    0.5298
   0.4692    0.4235    0.2259    0.6405
   0.0648    0.5155    0.5798    0.2091
   0.9883    0.3340    0.7604    0.3798
B = 0.7833
   0.6808
   0.4611
   0.5678
C = 0.7942    0.0592    0.6029    0.0503
D= 0
```

得到

```
f =    0          0.9689   -0.1059    0.0305   -0.0009
g =   1.0000    -1.6675   -0.2945   -0.0340    0.0055
```

在控制系统工具箱中有状态空间模型转换为传递函数模型的专门函数 ss2tf，运行

```
[f1, g1]=ss2tf(A, B, C, D)
```

可以得到同样的结果。在了解了它的原理后，可不必自编，直接调用即可。

【例 6.19】 系统的串联、并联和反馈

设 A，B 为两个单输入单输出的子系统，其传递函数为

$$W_A=\frac{10(0.5s+1)}{5s^2+2s+1},\quad W_B=\frac{4}{(s+1)s}$$

求将此两个子系统串联、并联和反馈后系统的传递函数。

解：■ 建模

多个子系统组成复合模型的系数矩阵的求法。

● 系统的串联

由图 6.18-1 所示，$Y_B= W_BU_B = W_BW_AU_A = WU$

复合系统的传递函数为 A，B，两系统传递函数的乘积

$$W(s) = W_A(s)\, W_B(s)$$

即

$$H(s)=\frac{f(s)}{g(s)}=\frac{f_A(s)f_B(s)}{g_A(s)g_B(s)}$$

在 MATLAB 中，多项式的相乘由卷积函数 conv 来实现（参看本书第 4.3 节），因此，其表示式为：

➢ f = conv(fA，fB)

➢ g = conv(gA，gB)

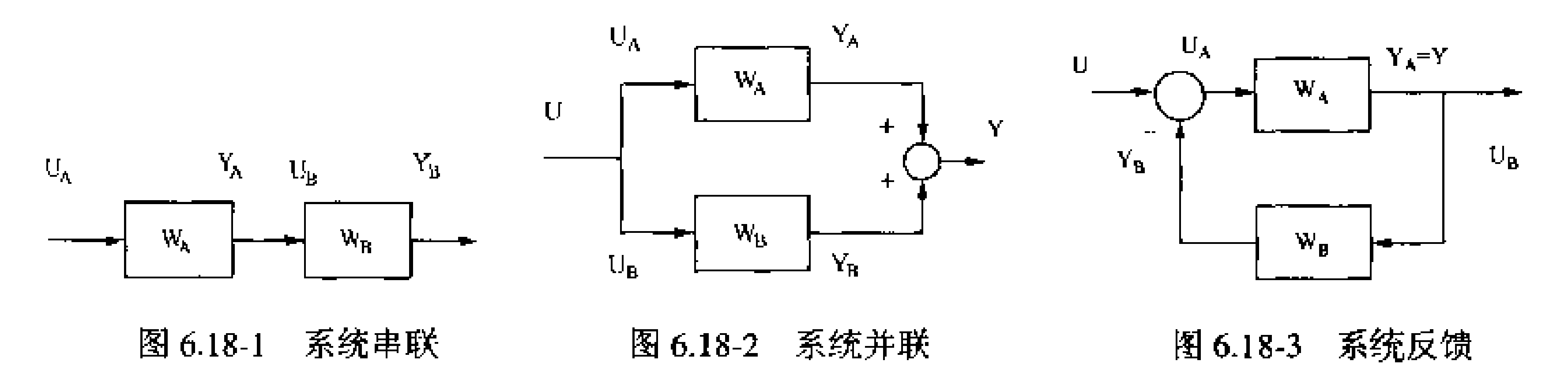


图 6.18-1　系统串联　　图 6.18-2　系统并联　　图 6.18-3　系统反馈

● 系统的并联

由图 6.18-2，可得

$$Y= W_AU + W_BU = (W_A+W_B)\, U = WU$$

复合系统的传递函数为 A，B，两系统传递函数的和为

$$W(s) = W_A(s) + W_B(s)$$

即

$$W(s)=\frac{f(s)}{g(s)}=\frac{f_A(s)g_B(s)+f_B(s)g_A(s)}{g_A(s)g_B(s)}$$

在 MATLAB 中多项式乘法用卷积函数 conv 实现，而多项式相加必须将短的系数向量前面补零，使两个多项式长度相同。在 4.3 节中，提供了完成多项式相加的函数 polyadd，因此可得

➢ f = polyadd(conv(fA，gB)，conv(fB，gA))

➢ g = conv(gA，gB)

● 系统的反馈

系统的连接方法如图 6.18-3。复合系统的传递函数

$$H(s)=\frac{W_A(s)}{1+W_A(s)W_B(s)}=\frac{f_A(s)g_B(s)}{f_A(s)f_B(s)+g_A(s)g_B(s)}$$

故 MATLAB 表达式为

➢ f=conv(fA，gB)

➢ g=polyadd(conv(fA，fB)，conv(gA，gB)

可以按照这些关系进行编程。

■ **MATLAB程序q619.m**

```
% 用传递函数法写出这两个系统的描述参数:
fA=[5, 10]; gA=[5, 2, 1];
fB=4; gB=[1, 1, 0];
% 两环节串联后合成的传递函数 fh1, gh1 为
fh1=conv(fA, fB);
gh1=conv(gA, gB);
disp('串联后的传递函数')
printsys(fh1, gh1, 's')
% 两环节并联后合成的传递函数 fh2, gh2 为
fh2=polyadd(conv(fA, gB), conv(gA, fB));
gh2=conv(gA, gB);
disp('并联后的传递函数')
printsys(fh2, gh2, 's')
% 将 B 环节放在负反馈支路上后合成的传递函数 fh3, gh3 为
fh3=conv(fA, gB);
gh3=polyadd(conv(fA, fB), conv(gA, gB));
disp('将 B 环节放在反馈支路上后的传递函数')
printsys(fh3, gh3, 's')
```

另外要存一个多项式相加函数程序 polyadd.m 供此程序调用，其内容为：

```
function r = polyadd(p, q)
% r = polyadd(p, q) 执行 r = p + q.
lp = length(p); lq = length(q); k = lp-lq;
if k>=0 r = p + [zeros(1, k), q ];
else r = [zeros(1, -k),  p ] + q; end
```

■ **程序运行结果**

键入 q619.m，得

串联后的传递函数

```
          20 s + 40
   ------------------------
   5 s^4 + 7 s^3 + 3 s^2 +  s
```

并联后的传递函数

```
num/den =
    5 s^3 + 35 s^2 + 18 s + 4
  ------------------------------
  5 s^4 + 7 s^3 + 3 s^2 +  s
```

将 B 环节放在反馈支路上后的传递函数

```
         5 s^3 + 15 s^2 + 10 s
  ------------------------------------
  5 s^4 + 7 s^3 + 3 s^2 + 21 s + 40
```

【例 6.20】 复杂系统的信号流图计算

遇到由大量环节交叉联接的系统，计算方法之一是靠综合点和分叉点的移动把系统归结为并联、串联和反馈；第二种方法是把任何复杂的结构，画成信号流图，用梅森公式来求解，但梅森公式的计算仍然是很麻烦的。如用 MATLAB 来辅助，就不宜直接用梅森公式，要采用另外规范的易于编程的方法，以便得出更简明的公式。

设信号流图中有 k_i 个输入节点，k 个中间和输出节点，它们分别代表输入信号 u_i（i=1，2，…，k_i）和系统状态 x_j（j=1，2，…，k）。信号流图代表它们之间的联结关系。用拉普拉斯算子表示后，任意状态 x_j 可以表为 u_i 和 x_j 的线性组合

$$x_j = \sum_{k=1}^{k} q_{jk} x_k + \sum_{i=1}^{k_i} p_{ji} u_i$$

用矩阵表示，可写成：

$$X = QX + PU$$

其中：X = [$x_1;x_2;\ldots;x_k$] 为 k 维状态列向量，$U=[u_1;u_2;\ldots;u_{k_i}]$ 为 k_i 维输入列向量，Q 为 $k\times k$ 阶的传输矩阵，P 为 $k\times k_i$ 阶的输入矩阵，Q 和 P 的元素 q_{ij} 和 p_{ij} 是各环节的传递函数。上式可写为

$$(I-Q)X = PU$$

由此得：

$$X = (I-Q)^{-1}PU$$

因此，系统的传递函数矩阵为 $H=(I-Q)^{-1}P$，这个简明的公式就等价于梅森公式。只要写出 P 和 Q，任何复杂系统的传递函数都可用这个简单的式子求出。

存在的困难是，传递函数是用常数及拉普拉斯算子组成的，可以用两个多项式系数向量来表示，但这个公式中用到的是普通的矩阵乘法和加法，无法应用于传递函数。

MATLAB 的工具箱解决了这个问题。它有两个途径：①利用符号运算 (Symbolic) 工具箱；②利用控制系统 (Control) 工具箱为线性系统建立对象类，将普通的矩阵乘法和加法扩展到这个数据类中去。先在本例中介绍第①条途径，第②条途径将在第 8 章中介绍。设系统的信号流图如图 6.19，求以 u 为输入，x_8 为输出的传递函数。

解：■ 建模

由图6.19列出方程为

$x_1=u$

$x_2=x_1-x_3-x_5$

$x_3=G_1x_2$

$x_4 = x_1+x_3-x_5$

$x_5 = G_2x_4$

$x_6 = x_3+x_5-x_7$

$x_7 = G_3x_6$

$x_8 = Kx_7$

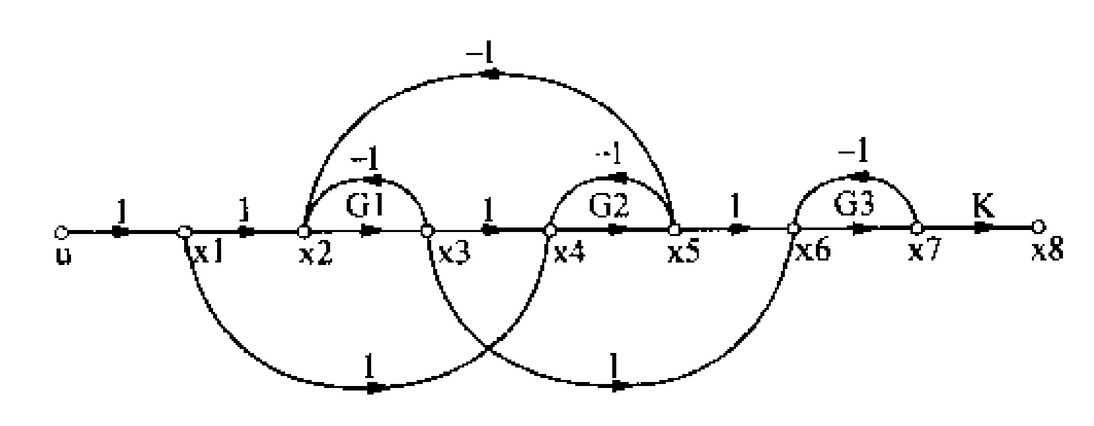


图 6.19　系统的信号流图

用矩阵表示，可写成

$$X = QX + PU$$

因 x_8 未出现在右端，$\boldsymbol{Q}$ 为 8 行 7 列，应在最后补上一个全零列，即为

$$\boldsymbol{Q}=\begin{bmatrix}0&0&0&0&0&0&0&0\\1&0&-1&0&-1&0&0&0\\0&G_1&0&0&0&0&0&0\\1&0&1&0&-1&0&0&0\\0&0&0&G_2&0&0&0&0\\0&0&1&0&1&0&-1&0\\0&0&0&0&0&G_3&0&0\\0&0&0&0&0&0&K&0\end{bmatrix}$$

$$\boldsymbol{P}=[1;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0];$$

以 u 为输入，$\boldsymbol{X}=[x_1; x_2; x_3; x_4; x_5; x_6; x_7; x_8]$。输出的系统传递函数为

$$\boldsymbol{H}=(\boldsymbol{I}-\boldsymbol{Q})^{-1}\boldsymbol{P}$$

它有 8 行，最后一行是求出的答案。

■ **MATLAB程序q620.m(要运行本程序，MATLAB必须装有Symbolic工具箱)**

```
syms G1 G2 G3 K s          % 定义字符自变量
Q(3, 2)=G1;                % 采用字符矩阵，第一条赋值语句右端必须是字符变量
Q(2, 1)=1;Q(2, 3)=-1;Q(2, 5)=-1;  % 列出连接矩阵
Q(4, 3)=1;Q(4, 1)=1;Q(4, 5)=-1;
Q(5, 4)=G2;
Q(6, 3)=1;Q(6, 5)=1;Q(6, 7)=-1;
Q(7, 6)=G3;  Q(8, 7)=K;
Q(:, end+1)=zeros(max(size(Q)), 1)      % 加一个全零列，补成方阵
B=[1; 0; 0; 0; 0; 0; 0; 0];
I=eye(size(Q));
W=(I-Q)\B             % 求出完整的传递矩阵
W8 = W(8)             % x8 为输出的传递函数为其第八项 W(8)
```

```
pretty(W8)            % 给出便于阅读的形式
```

■ 程序运行结果

```
W8 =G3*K*(2*G2*G1+G1+G2)/(2*G2*G1+2*G2*G1*G3+G3+G1+1+G2+G2*G3+G1*G3)
```

不难看出，它和梅森公式的结果完全相同。

如果把 G1，G2，G3 作为自变量 s 的因变量，不让它们出现在程序中，则把其中的三条语句改为

```
Q(3, 2)=s/(s+1); Q(5, 4)=3/(s+4); Q(7, 6)=(s^2+5*s+6);
```

运行此程序的结果为（此处用公式排版，与 MATLAB 命令窗中显示的略有不同）

$$W_8=\frac{K(s^4+18s^3+74s^2+93s+18)}{2s^4+161s+28s^3+111s^2+49}$$

这就是系统的传递函数。可以看到，幂次排列还有些乱。究竟计算机不够聪明，最好的分工是计算机做烦琐计算，而人则参与一些思维判断。

【例 6.21】 连续系统状态方程的零输入响应

线性连续系统在输入信号为零时的状态方程表示式为

$$\dot{x}=Ax$$

其中 x 为 $n\times1$ 的列向量，而 A 为一个 $n\times n$ 阶的常系数方阵，求零输入响应。

解：■ 建模

按线性方程理论，此齐次方程的解为

$$x(t)=\mathrm{e}^{A(t-t_0)}x(t_0),\qquad t\geqslant t_0$$

其中，指数矩阵 $\mathrm{e}^{A(t-t_0)}$ 是 $n\times n$ 阶的，称为状态转移矩阵 $\Phi(t-t_0)$，它把 t_0 时的状态变量 $x(t_0)$，变换为 t 时刻的状态变量 $x(t)$。对定常系统，$\Phi(t-t_0)=\mathrm{e}^{A(t-t_0)}$。

为了与 MATLAB 的符号衔接，后面将把 Φ 改为 F，初始条件 $x(0)$ 改为 x0，它是 n 维的列向量。从上式可以看出，求状态方程解的问题，主要是求指数矩阵。它有很多种计算方法，但通常都很繁，用手工算，三阶以上就很难了，按矩阵指数函数的定义有：

$$\mathrm{e}^{At}=I+At+\frac{1}{2!}A^2t^2+\frac{1}{3!}A^3t^3+\cdots+\frac{1}{k!}A^kt^k+\cdots$$

前面多次用过 MATLAB 的指数函数 exp，那是对标量的，即使涉及矩阵，也是对矩阵的各个元素作运算，即元素群运算。此处要把方阵 At 作为一个整体求指数函数，就需要调用 MATLAB 矩阵指数函数 expm。注意它与 exp 函数不同，多了一个 m，表示矩阵指数函数，其同类的函数还有 expm1，expm2，expm3，都是用来算矩阵指数函数的，只是用的方法不同而已。

如果 A 是 n×n 阶，则 expm(At)也是 n×n 阶，这时如果要算一系列的 t 值所对应的 expm(At)，就不可能像标量指数那样用元素群运算方法了，必须用 for 循环。不仅如此，如果 t 的长度为 nt，则状态转移矩阵 F(t)=expm(At)将是一个 n×n×nt 的三维矩阵，需要用 MATLAB 中高维矩阵的概念来解决问题。举以下数字例来说明。

设 A＝[−2，1;−17，−4]，x0＝[3 ;4]; 求上述线性方程的状态转移矩阵 F(t)及 x 的解。

■ MATLAB程序q621.m

```
% 用多维矩阵求状态转移矩阵及齐次状态方程解
A=[-2, 1;-17, -4];         % 输入状态方程系数矩阵
x0=[3;4];                  % 输入初始条件
```

```
t=0:.02:3; Nt=length(t);            % 设定自变量数组并确定其长度
F=zeros(2, 2, Nt); x=zeros(2, Nt); % 状态转移矩阵 F 及状态变量初始化
for k=1:Nt                                         % 对对间循环
   F(:, :, k)=expm(A*t(k));          % 计算各时刻的状态转移矩阵 F
end
z=reshape(F, [4, Nt]);              % 把 F 变为二维矩阵以便绘图
% 第一张图是系统状态转移矩阵，plot 语句只接受二维变量
z(1, :)=F(1, 1, :), z(2, :)=F(2, 1, :)
z(3, :)=F(1, 2, :), z(2, :)=F(2, 2, :)
subplot(2, 1, 1)
plot(t, z(1, :), '-.', t, z(2, :), ':', t, z(3, :), '-', t, z(4, :), '—'),
grid,
legend('F(1, 1)', 'F(2, 1)', 'F(1, 2)', 'F(2, 2)')
title('系统的状态转移矩阵')
for k=1:Nt                                  % 对时间循环，求各点状态变量
% 矩阵乘法只能用于二维，因此对每一时刻的 F，用 squeeze 函数缩去长度为 1 的第三维
   x(:, k)=squeeze(F(:, :, k)) *x0;
end
% 第二张图是在给定初始条件下的输出
subplot(2, 1, 2), plot(t, x(1, :), '-', t, x(2, :), '—'), grid
legend('x(1, :)', 'x(2, :)')
title('系统输出状态变量')
```

■ 程序运行结果

输出的数据此处予以省略，只把输出的曲线表示在图 6.20 中。曲线的标注用了 legend 命令。在这个过程中，为了弄清各条曲线所代表的变量，还必须查看各变量的数据。

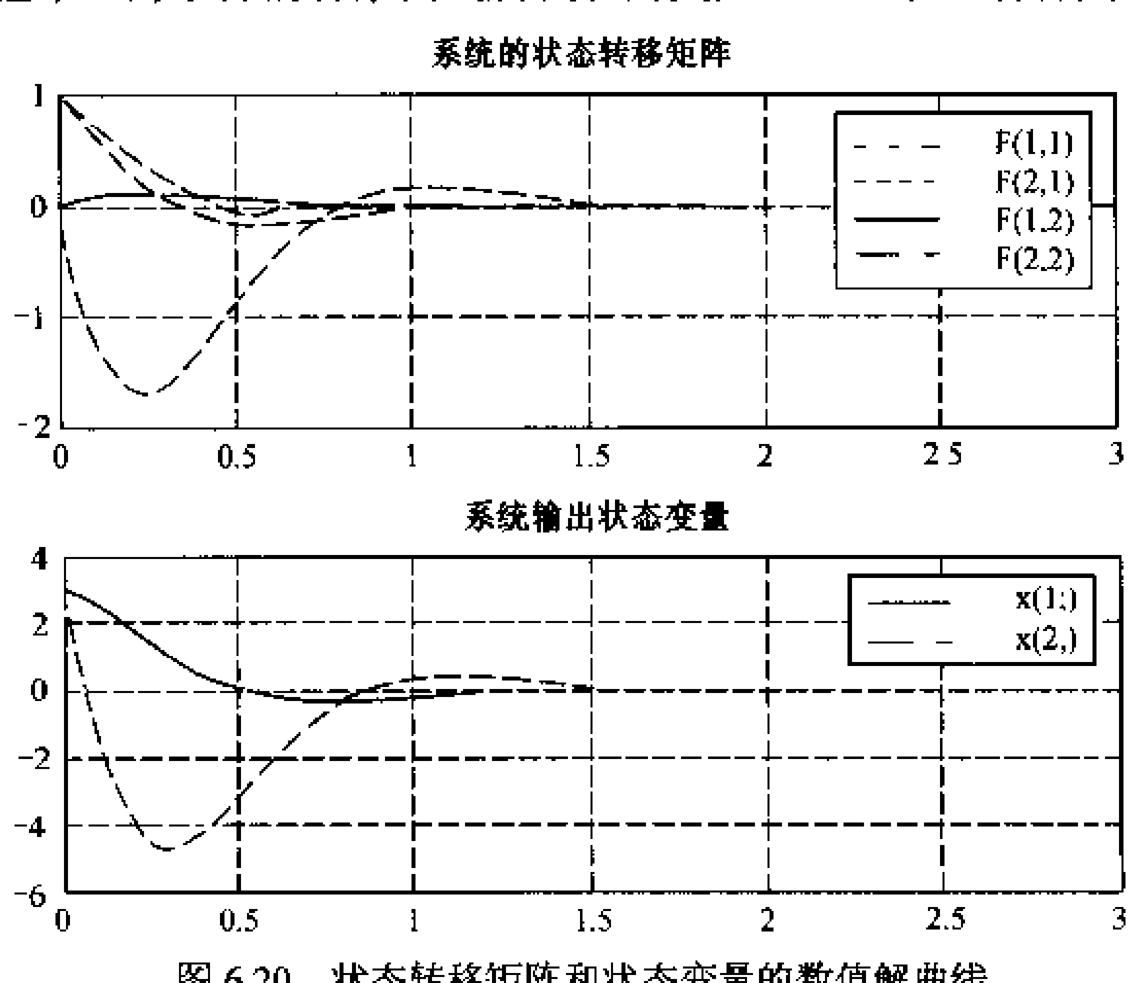


图 6.20 状态转移矩阵和状态变量的数值解曲线

【例 6.22】 离散系统状态方程的响应

设线性离散系统的状态方程为

$$x(k+1)=Ax(k)+Bu(k)$$

其中 x 为 2×1 阶的列向量，A，B 分别为 2×2 阶和 2×1 阶的常系数阵，$A=\begin{bmatrix}0.5 & 0\\0.25 & 0.25\end{bmatrix}$，$B=\begin{bmatrix}1\\0\end{bmatrix}$，初始条件为 $x(0)=\begin{bmatrix}-1\\0.5\end{bmatrix}$，输入信号为幅度 0.5 的阶跃函数（从 k=1 时刻作用），求前 10 步的解 x。

解: ■ 建模

此离散方程的矩阵形式为

$$\begin{bmatrix}x_1(k+1)\\x_2(k+1)\end{bmatrix}=\begin{bmatrix}0.5 & 0\\0.25 & 0.25\end{bmatrix}\begin{bmatrix}x_1(k)\\x_2(k)\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix}u(k)$$

已知 $x(0)$和 $u(0)$就可求出 k=1 时的 $x(1)$，再依次递推求 $x(2)$，…，$x(10)$。在程序中可用循环语句来完成。总的来说，离散系统的计算比连续系统简单得多，只要作四则矩阵运算即可。当然矩阵阶数较大和计算步数较多时，算起来也很容易出错，用计算机辅助很有必要。

编程时要注意 MATLAB 中变量下标不允许为零，初始点的下标只能取 1，第 n 步的 x，应标为 x (n+1)，

另外要注意 x 数组的方向。x 是一个二维数组，它的各行表示不同的状态，而各列则表示不同的时间。在用绘图命令时，要注意如何把时间原点调整到以零为起点。另外，本程序有一定的通用性，对任何单输入的 n 阶离散状态方程系统都能应用。

■ MATLAB程序q622.m

```
clear
A = input('离散状态方程系数 A=(N*N 方阵)  ');
B = input('离散系统状态方程系数 B=(N*1 列阵)  ');
x0 = input('初始状态变量 x0=(N*1 数组)  ');
n = input('要计算的步数 n=  ');
u = input('输入信号 u=（长度为 n 的数组）  ');
x(:, 1)= x0;          % 因 MATLAB 下标不能取 0, x(1)对应于 k=0 处的 x
for i=1:n                        % 递推计算 n 次
   x(:, i+1)= A*x(:, i)+B*u(i);
end
% 画出前两个状态变量的曲线，注意如何使下标还原至 k=0 为起点
subplot(2, 1, 1), stem([0:n], x(1, :))
subplot(2, 1, 2), stem([0:n], x(2, :))
```

■ 程序运行结果

运行程序并按提示输入：

```
A=  [0.5, 0;0.25, 0.25]
B=  [1;0]
x0=  [-1;0.5]
n=  10
```

```
u=  [0, 0.5*ones(1, n-1)]
```

得出的曲线如图 6.21 所示。

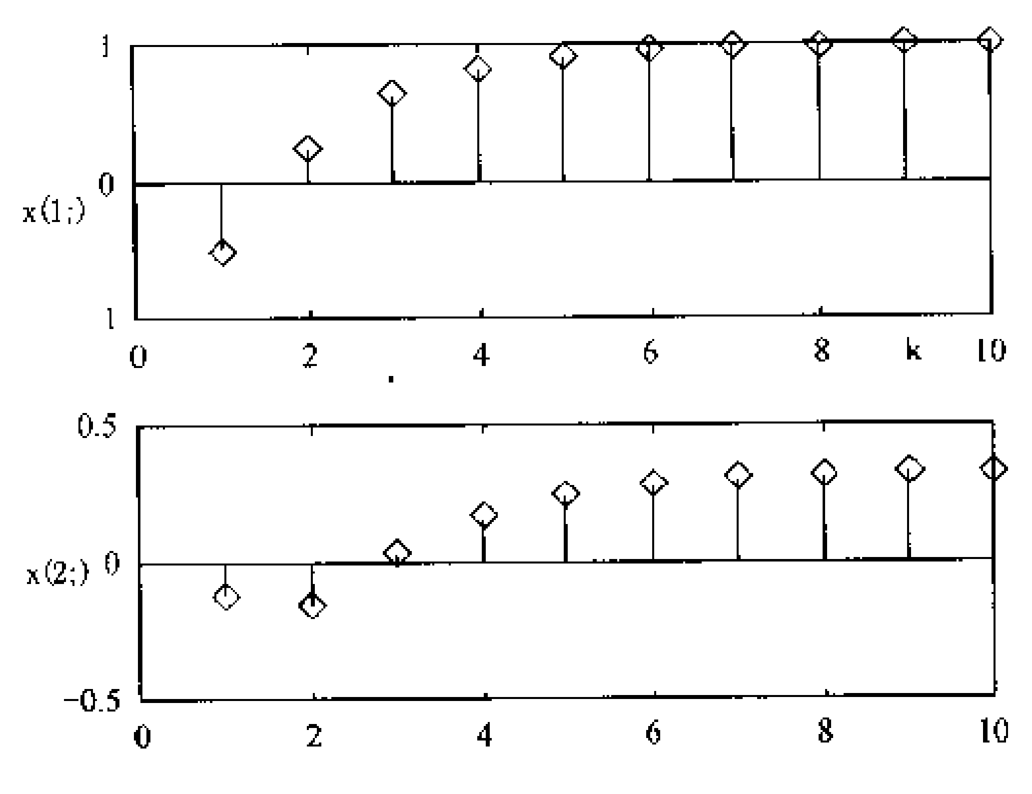


图 6.21　离散系统的状态方程的解

第 7 章　MATLAB 在数字信号处理中的应用

数字信号处理是一门理论与实践紧密结合的课程。做大量的习题和上机实验，有助于进一步理解和巩固理论知识，还有助于提高分析和解决实际问题的能力。过去用其他算法语言，实验程序复杂，在有限的实验课时内所做的实验内容太少。MATLAB 强大的运算和图形显示功能，可使数字信号处理上机实验效率大大提高。特别是它的频谱分析和滤波器分析与设计功能很强，使数字信号处理工作变得十分简单、直观。本章结合数字信号处理的典型例题说明用 MATLAB 进行数字信号处理实验的编程方法与技巧。

在第 6 章中已经分析了线性时不变系统的一般数学模型和求解方法。读者在学习本章之前，必须先阅读这些内容。本章是它的深化，主要讨论离散线性时不变系统的分析理论和方法。在本书前面各章中，几乎不用 MATLAB 的工具箱，其目的是，（1）避免读者对原理的忽视；（2）提高读者的 MATLAB 编程能力。随着研究问题的日益复杂，再拒绝工具箱就不合适了。

本章前三节为数字信号处理的基础理论，为了后面编写程序方便，先介绍本章常用的几个工具箱函数。然后，在各节中结合例题介绍其他函数。为了便于读者查阅，将 MATLAB 信号处理工具箱函数分类列于第 7.7 节中。

7.1　时域离散信号的产生及时域处理

时域离散信号用 $x(n)$表示，时间变量 n（表示采样位置）只能取整数。因此，$x(n)$是一个离散序列，以后简称序列。序列适合计算机存储与处理。在本书 6.3 节中已做了初步介绍，本节将深入讨论用 MATLAB 对序列的运算。

由 6.3 节可知，在 MATLAB 中，向量 $\boldsymbol{x}$ 的下标只能从 1 开始，不能取零或负值，而 $x(n)$中的时间变量 n 则完全不受限制。因此，向量 $\boldsymbol{x}$ 的下标不能简单地看做时间变量 n，用一个向量 $\boldsymbol{x}$ 不足以表示序列值 $x(n)$。必须再用另一个等长的定位时间变量 $\boldsymbol{n}$。$\boldsymbol{x}$ 和 $\boldsymbol{n}$ 同时使用才能完整地表示一个序列，只有当序列的时间变量正好从 1 开始时才可省去 $\boldsymbol{n}$。

由于 n 序列是按整数递增的，可简单地用其初值 ns 决定，因为它的终值 nf 取决于 ns 和 x 的长度 length(x)，故可写成：

➢ n = [ns:nf]

或　n = [ns: ns+length(x)−1]

6.3 节中已介绍了常用离散序列的生成，这里把几个常用离散序列写成子程序，以便调用。

■ 单位脉冲序列 $\delta(n-n_0)$的生成函数 impseq

➢ function [x，n] = impseq(n0，ns，nf)

n = [ns:nf]; x = [(n−n0) = = 0];

序列的起点为 ns，终点为 nf，在 n = n0 点处生成一个单位脉冲。

■ 单位阶跃序列 $u(n-n_0)$的生成函数 stepseq

➢ function [x，n] = stepseq(n0，ns，nf)

n = [ns:nf]; x = [(n−n0) >= 0];

序列的起点为 ns， 终点为 nf，在 n = n0 点处生成单位阶跃。

本节着重讨论序列的运算，通过这些运算，可以产生更复杂的序列。表 7.1 给出了一些常用的运算规则，请读者结合例题思考。

表 7.1 一些常用的序列运算及其 MATLAB 表述

运 算	数学形式	MATLAB 表述
两序列相加	$y(n)=x_1(n)+x_2(n)$	将两序列时间变量延拓至同长，x1 和 x2 成为 x1a 和 x2a，然后逐点相加求 y = x1a+x2a
两序列相乘(加窗)	$y(n)=x_1(n)x_2(n)$	将两序列时间变量延拓至同长，x1 和 x2 成为 x1a 和 x2a，然后逐点相乘求 y = x1a.*x2a
序列累加(与积分类似)	$y(n)=\sum_{i=ns}^{n}x(i)$	y = cumsum(x)
右移位 m	$y(n)=x(n-m)$	y = x; ny = nx-m
对 n = m 点折叠	$y(n)=x(-(n-m))$	y = fliplr(x); ny = fliplr(-(nx-m)) fliplr 为左右翻转函数
长 M 的周期延拓	$y(n)=x((n))_M$	ny = nsy:nfy; y = x(mod(ny， M)+1);
两序列的卷积	$y(n)=x_1(n)\otimes h_1(n)$	y = conv(x1， x2)
序列的能量	$E=\sum_{n=ns}^{nf}x(n)x^*(n)$	E = x*conj(x)' 或 E = sum(abs(x).^2)
两序列的相关	$y(m)=\sum_{n=ns}^{nf}x_1(n)x_2(n-m)$	y = xcorr(x1， x2)
序列的傅立叶变换		X = fft(x， N)
序列通过线性系统	差分方程求解	y = filter(B， A， x)

表 7.1 中所用的大部分函数都是 MATLAB 基本部分的函数，只有 filter 函数是信号处理工具箱函数。其用法如下：

■ filter 一维数字滤波函数。

➢ y =filter(B， A， x) 对向量 x 中的数据进行滤波处理，即差分方程求解，产生输出序列向量 y。B 和 A 分别为数字滤波器系统函数 H(z)的分子和分母多项式系数向量。

$$H(z)=\frac{B(z)}{A(z)}=\frac{B(1)+B(2)z^{-1}+\cdots+B(N)z^{-(N-1)}+B(N+1)z^{-N}}{A(1)+A(2)z^{-1}+\cdots+A(N)z^{-(N-1)}+A(N+1)z^{-N}}$$

filter 函数还有多种调用方式，请用 help 语句查阅。filter2 为二维数字滤波函数。

【例 7.1】 序列的相加和相乘

为了说明表中前两项的算法，给出两个序列 $x_1(n)$和 $x_2(n)$。

x1 = [0， 1， 2， 3， 4， 3， 2， 1， 0]； n1 = [-2:6,]；

x2 = [2， 2， 0， 0， 0， -2， -2]， n2 = [2:8]；

现在要求它们的和 ya 及乘积 yp。

解： ■ MATLAB程序q701. m

```
x1=[0,1,2,3,4,3,2,1,0];ns1=-2;          % 给定 x1 及 ns1
x2=[2,2,0,0,0,-2,-2]; ns2=2;            % 给定 x2 及 ns2
```

```
nf1=ns1+length(x1)-1; nf2=ns2+length(x2)-1;
ny= min(ns1,ns2):max(nf1,nf2);                  % y(n)的时间变量
xa1 = zeros(1,length(ny)); xa2 = xa1;           % 延拓序列初始化
xa1(find((ny>=ns1)&(ny<=nf1)==1))=x1;           % 给 xa1 赋值 x1
xa2(find((ny>=ns2)&(ny<=nf2)==1))=x2;           %  给 xa2 赋值 x2
ya = xa1 + xa2                                  % 序列相加
yp = xa1.* xa2                                  % 序列相乘
subplot(4,1,1), stem(ny,xa1,'.')                % 绘图
subplot(4,1,2), stem(ny,xa2,'.')
line([ny(1),ny(end)],[0,0])                     % 画 x 轴
subplot(4,1,3), stem(ny,ya,'.')
line([ny(1),ny(end)],[0,0])
subplot(4,1,4), stem(ny,yp,'.')
line([ny(1),ny(end)],[0,0])
```

■ **程序运行结果**

```
ny =-2 -1 0 1 2 3 4 5 6 7  8
xa1 =0  1 2 3 4 3 2 1 0  0  0
xa2= 0  0 0 0 2 2 0 0  0 -2 -2
ya = 0  1 2 3 6 5 2 1 0 -2 -2
yp = 0 0 0 0  8 6 0 0 0  0  0
```

相应的图形见图 7.1。

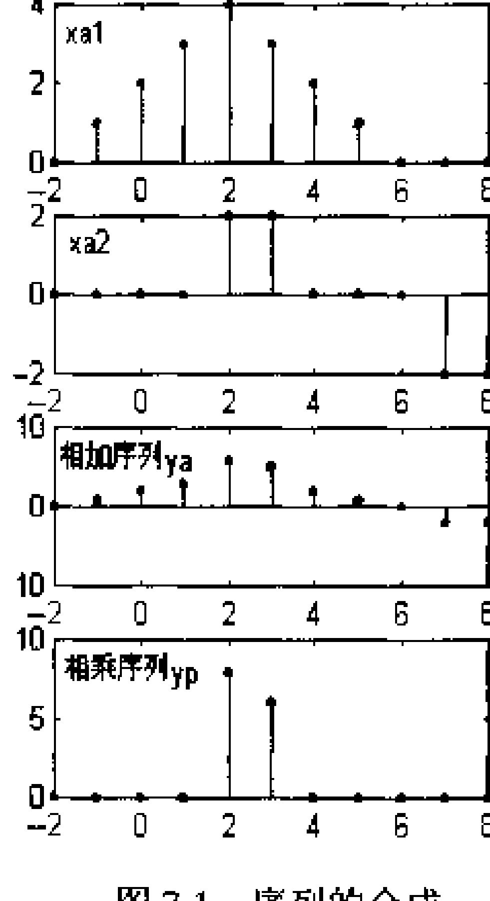


图 7.1 序列的合成

可以看出，延拓的序列长度覆盖了 n1 和 n2 的范围，这样才能把两序列的时间变量对应起来，然后进行对应元素的运算。

【例 7.2】 序列的合成和截取

用例 6.13 的结果编写产生矩形序列 $R_N(n)$的程序。序列起点为 n_0，矩形序列起点为 n_1，长度为 N（n_0，n_1，N 由键盘输入）。并用它截取一个复正弦序列 $e^{j\frac{\pi}{8}n}$，最后画出波形。

解：■ 建模

一个矩形序列可看成两个阶跃序列之差。即

$$x_1(n)=R_N(n)=U(n-n_1)-U(n-n_1-N)$$

本程序中也巧妙地利用 MATLAB 逻辑关系运算产生了矩形序列 $x_2(n)$。而用矩形序列截取任何序列相当于两序列的元素群相乘 x1.*x，也称为加窗运算。序列的合成和截取实际上就是相加和相乘。由于本题两序列时间变量本来就一致，所以程序可以简单些。

■ **MATLAB程序q702.m**

```
clear; close all
n0=input('输入序列起点:n0=');
N=input('输入序列长度:N=');
n1=input('输入位移:n1=');
n=n0:n1+N+5;                              % 生成时间变量数组
u=[(n-n1)>=0];                            % 产生单位阶跃序列（u(n-n1)）
x1=[(n-n1)>=0]-[(n-n1-N)>=0]              % 用阶跃序列之差产生矩形序列
```

```
x2=[(n>=n1)&(n<(N+n1))];                    % 用逻辑式产生矩形序列
x3=exp(j*n*pi/8). *x2;                      % 对复正弦序列加矩形窗（元素群乘）
subplot(2,2,1);stem(n,x1,'.');
xlabel('n');ylabel('x1(n)');                % 标注
axis([n0,max(n),0,1]);                      %定坐标范围
subplot(2,2,3);stem(n,x2,'.');
xlabel('n');ylabel('x2(n)');                % 标注
axis([n0,max(n),0,1]);                      %定坐标范围
subplot(2,2,2);stem(n,real(x3),'.');
xlabel('n');ylabel('x3(n)的实部');          % 标注
line([n0,max(n)],[0,0]);                    % 画横轴
axis([n0,max(n),-1,1]);                     % 定坐标范围
subplot(2,2,4);stem(n,imag(x3),'.');
xlabel('n');ylabel('x3(n)的虚部');          % 标注
line([n0,max(n)],[0,0]);                    % 画横轴
axis([n0,max(n),-1,1]);                     % 定坐标范围
```

■ 程序运行结果

按提示输入n0=-6，N=15及n1=3，结果如图7.2所示。

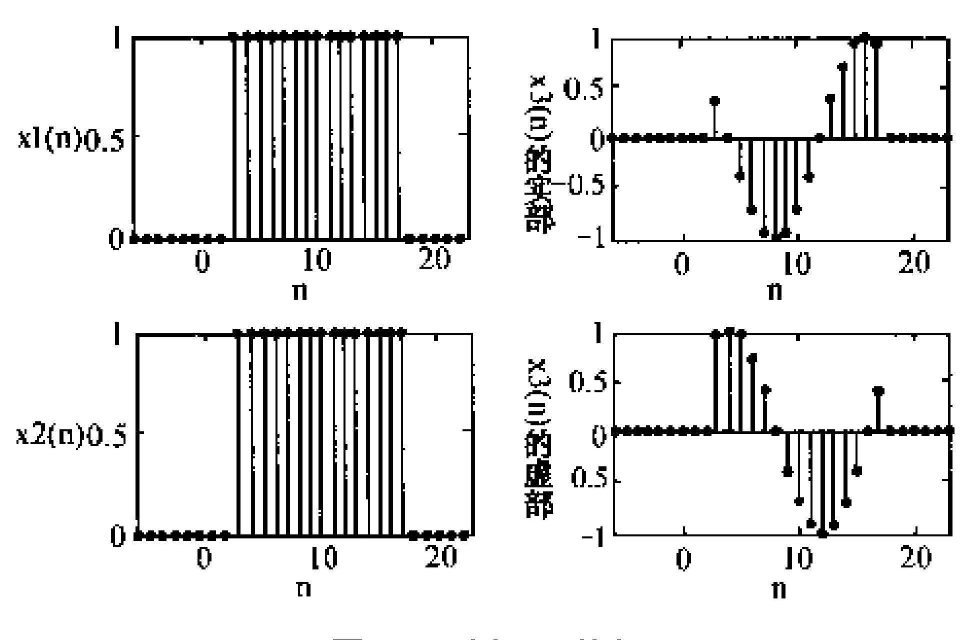


图 7.2　例 7.2 的解

可以看出，在这个程序中，标注、画横轴、定坐标范围等占了很多篇幅，而且在各个程序中大同小异。它并未给出更多新知识，因此，在以后的程序中将被省略。不过本书的软盘将提供完整的程序。

【例 7.3】 序列的移位和周期延拓运算

已知 $x(n)=0.8^n R_8(n)$，利用 MATLAB 生成并图示 $x(n)$，$x(n-m)$，$x((n))_8 R_N(n)$（$x((n))_8$ 表示 $x(n)$以 8 为周期的延拓）和 $x((n-m))_8 R_N(n)$，其中 N=24，m 为一个整常数，0<m<N。

解：■ 建模

取序列的观察区间为 24。利用 MATLAB 的矩阵运算和冒号运算可使周期延拓程序简短明了。假如取三个延拓周期，则

```
x=[1 2 3 4];
y=x'*ones(1, 3);
```

```
=1      1      1
 2      2      2
 3      3      3
 4      4      4
y1=(y(:))'
 =[1 2 3 4 1 2 3 4 1 2 3 4]
```

另一种更好的方法是采用 MATLAB 求余函数 mod，y = x(mod(n, M)+1)可实现对 x(n) 以 M 为周期的周期延拓，其中求余后加 1 是因为 MATLAB 向量下标从 1 开始，这样使程序更为简洁。

■ MATLAB程序q703.m

```
clear;close all
N=24;M=8;
m=input('输入移位值: m=');
if    (m<1|m>=N-M+1)                          % 检验输入参数 m 是否合理
      fprintf('输入数据不在规定范围内! '); break
end
      n=0:N-1;
      x1=(0.8).^n;  x2=[(n>=0)&(n<M)];        % 产生 x(n)
      xn=x1.*x2;
      xm=zeros(1,N);                          % 设定 xm 的长度
      for k=m+1:m+M
          xm(k)=xn(k-m);
      end
      xc=xn(mod(n,8)+1);                      % 产生 x(n)的周期延拓
      xcm=xn(mod(n-m,8)+1);                   % 产生 x(n)移位后的周期延拓
...
```

程序下段略。

用 stem(n, xn, '.')，stem(n, xm, '.')，stem(n, xc, '.')，stem(n, xcm, '.') 四条绘图语句画出四个图形，图形分割及标注程序从略，坐标范围均取 axis([0, length(n), 0, 1])。

■ 程序运行结果

如图 7.3 所示。

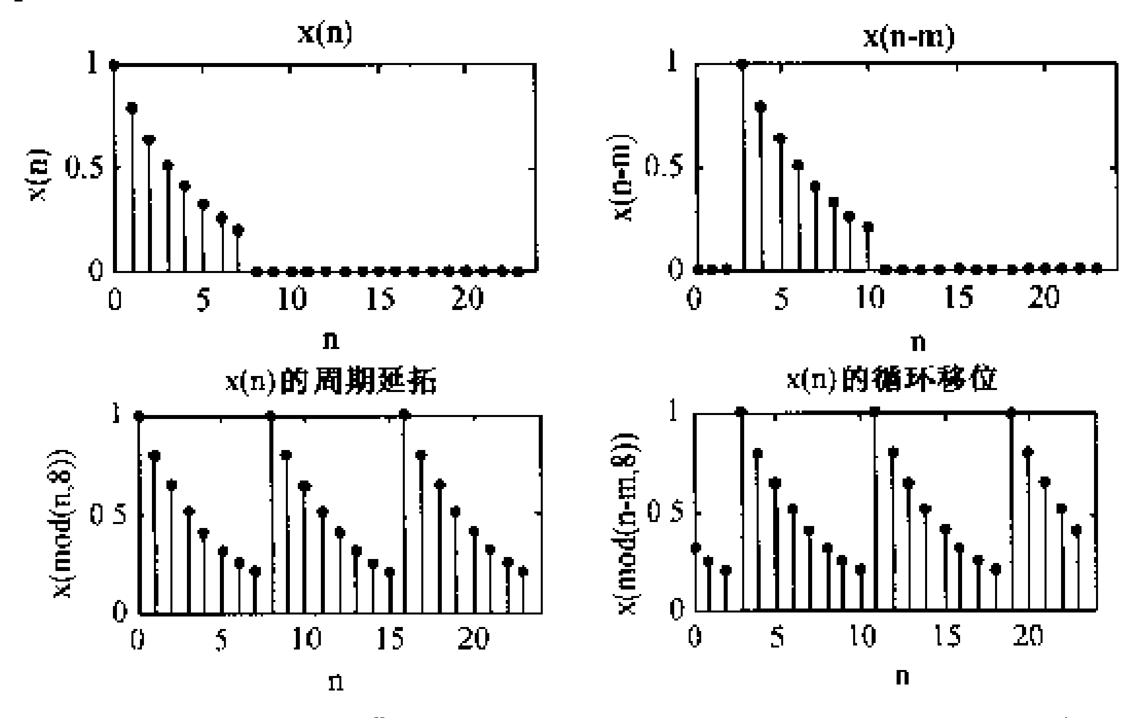


图 7.3　$x(n)=0.8^n R_8(n)$ 及其移位、周期延拓和循环移位序列

【例 7.4】 离散系统对几种常用信号的响应

给定因果稳定线性时不变系统的差分方程

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

对下列输入序列 $x(n)$，求出系统的输出序列 $y(n)$。

（1） $x(n)=\delta(n)$

（2） $x(n)=\delta(n-10)$

（3） $x(n)=u(n)$

（4） $x(n)=R_{32}(n)$

（5） $x(n)=\mathrm{e}^{\mathrm{j}\frac{\pi}{8}n}R_{32}(n)$

解：■ 建模

本题的计算原理见例 6.14，在这里用工具箱函数 filter 来解。如果已知系统函数 $H(z)=B(z)/A(z)$，则 filter 函数可求出系统对输入信号 $x(n)$的响应 $y(n)$。

y = filter(B, A, x)

由差分方程可得到 $H(z)$的分子和分母多项式系数向量

$\boldsymbol{B} = [b_0, b_1, b_2, b_3, \cdots, b_m]$;

$\boldsymbol{A} = [a_0, a_1, a_2, a_3, \cdots, a_m]$;

$\boldsymbol{x}$ 为输入信号向量。程序 q704.m 中，$\boldsymbol{B}$ 和 $\boldsymbol{A}$ 为 6 阶低通数字滤波器的差分方程系数矩阵。该滤波器 3dB 截止频率为 0.2π，对本题中五种输入信号的响应 $y_1(n)$，$y_2(n)$，$y_3(n)$，$y_4(n)$ 和 $y_5(n)$，如图 7.4 所示。从图中可以看出低通滤波器对各种信号的暂态响应过程和稳态响应趋势。由 $y_1(n)$和 $y_2(n)$可看出系统的时不变性。

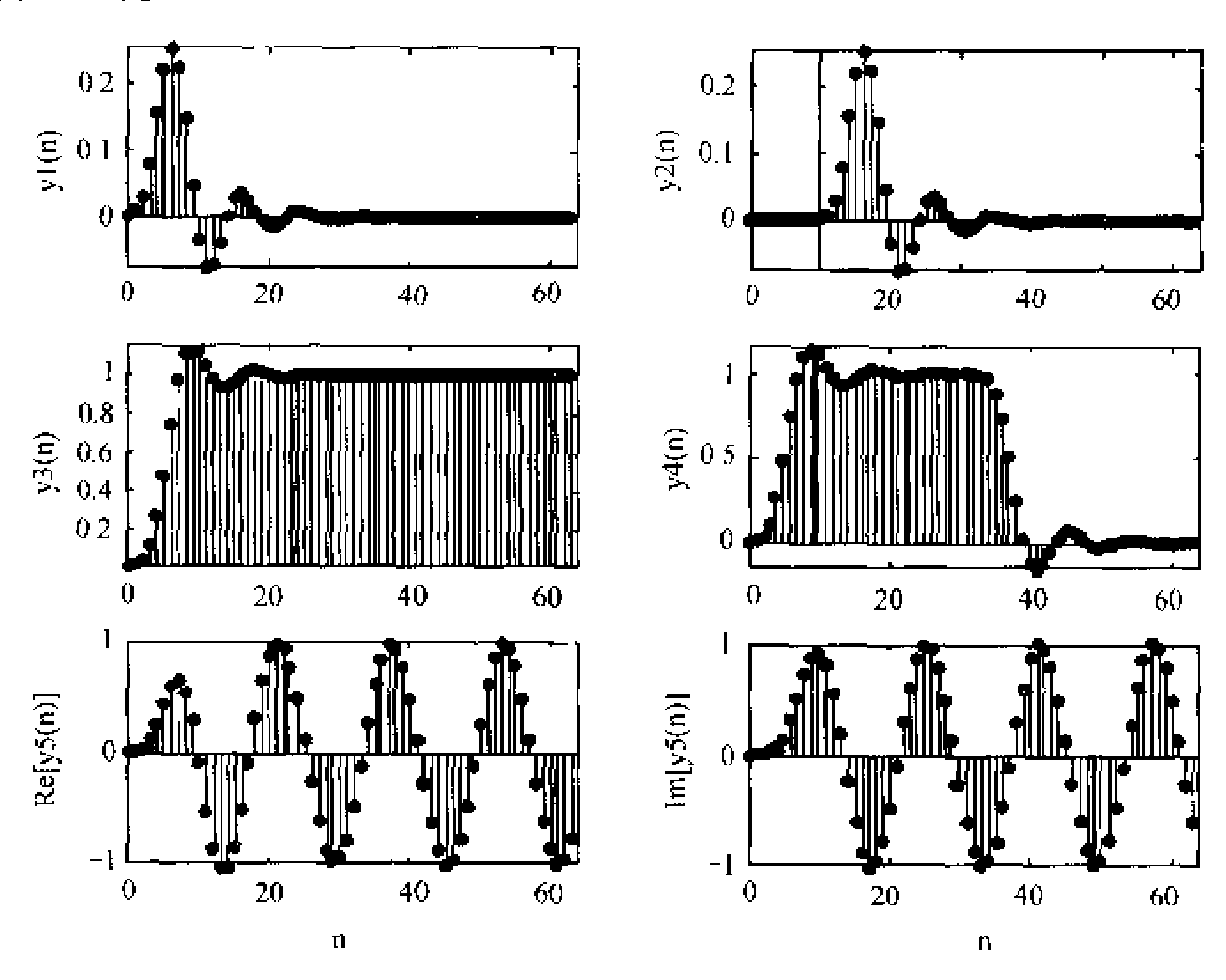


图 7.4　六阶低通数字滤波器对例 7.4 中五种信号的响应

■ MATLAB程序q704.m

```
%求时域离散系统对常用序列的响应
clear; close all;
N=64; n=0: N-1; m=10;
B=[1, 6, 15, 20, 15, 6, 1] *0.0007378;
A=[1.0000, -3.1836, 4.6223, -3.7795, 1.8136, -0.4800, 0.0544];
x1=[n==0];              %产生输入信号 x1(n)
y1=filter(B, A, x1);    %系统对 x1(n) 的响应 y1(n)
x2=[(n-m)==0];          %产生输入信号 x2(n)
y2=filter(B, A, x2);    %系统对 x2(n) 的响应 y2(n)
x3=[n>=0];
y3=filter(B, A, x3);
x4=[(n>=0)&(n<32)];
y4=filter(B, A, x4);
x5=exp(j*pi*n/8);
y5=filter(B, A, x5);
……
```

程序下段略。

用 stem(n, y1, '.'), stem(n, y2, '.'), stem(n, y3, '.'), stem(n, y4, '.'), stem(n, real(y5), '.'), stem(n, imag(y5), '.') 6 条绘图语句画出 6 个图形。图形分割及标注程序从略，坐标范围均取 axis([0, N, −1, 1])。

■ 程序运行结果

如图 7.4 所示。

【例 7.5】 系统线性性质验证

设系统差分方程为

$$y(n)=x(n)+0.8y(n-1)$$

要求用程序验证系统的线性性质。

解：分别产生输入序列 $x_1(n)=0.8^n R_{32}(n)$， $x_2(n)=\delta(n-20)$， $x_3(n)=5x_1(n)+3x_2(n)$。

计算并图示系统对 $x_1(n)$， $x_2(n)$ 和 $x_3(n)$ 的响应序列 $y_1(n)$， $y_2(n)$ 和 $y_3(n)$。

计算并图示 $y(n)=5y_1(n)+3y_2(n)$，观察 $y(n)$ 与 $y_3(n)$ 的波形。判断其正确性，并用线性系统理论解释。

本题编程的重点是从输出波形上验证线性系统的性质（齐次性和可加性）。

$$T[k_1x_1(n)+k_2x_2(n)]=k_1T[x_1(n)]+k_2T[x_2(n)]$$

其中，$T[x(n)]$表示线性系统对输入信号 $x(n)$的变换，即系统响应 $y(n)=T[x(n)]$。

■ MATLAB程序q705.m

```
clear; close all;
N=64; n=0: N-1; m=20;
B=1; A=[1, -0.8];          % 设定系统参数
x1=0.8.^n;                 % 产生输入信号 x1(n)
x=[(n>=0)&(n<32)];         % 用矩形窗截取
```

```
x1=x1.*x;
y1=filter(B,A,x1);          % 对 x1(n)的响应 y1(n)
x2=[(n-m)==0];
y2=filter(B,A,x2);          % 对 x2(n)的响应 y2(n)
x3=5*x1+3*x2;
y3=filter(B,A,x3);          % 对 5x1(n)+3x2(n)的响应 y3(n)
y=5*y1+3*y2;                % y(n)=5y1(n)+3y2(n)
```

程序下段略。

用 stem(n, y1, '.')，stem(n, y2, '.')，stem(n, y3, '.')，stem(n, y, '.') 4 条绘图语句画出 4 个图形。图形分割及标注程序从略。

■ **程序运行结果**

如图 7.5 所示。从图中可以看出 y(n) = y3(n)，也可在命令窗中键入 y 和 y3，使之显示出 y(n)和 y3(n)的全部数据，证明它们相同从而满足线性性质。

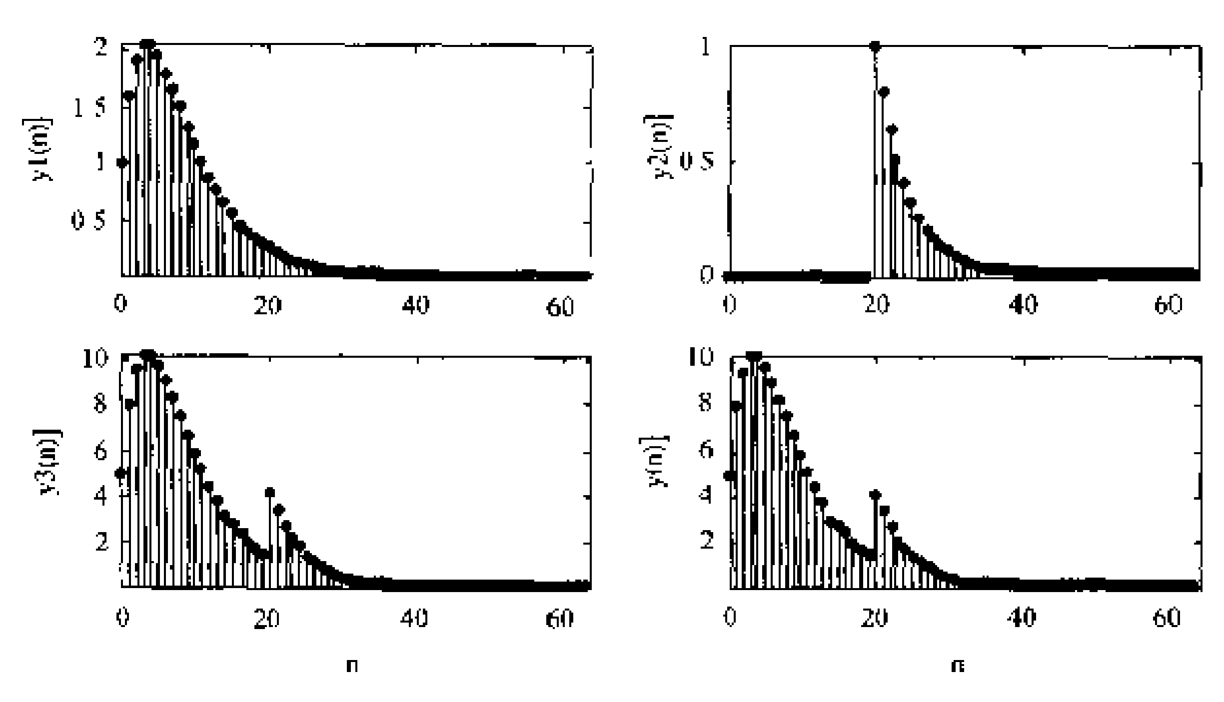


图 7.5　程序 q705.m 的执行结果

【例 7.6】 离散序列的卷积计算

计算下列卷积，并图示各序列及其卷积结果。

（1）　$y_1(n)=x_1(n)^*h_1(n)$，$x_1(n)=0.9^n R_{20}(n)$，$h_1(n)=R_{10}(n)$。

（2）　$y_2(n)=x_2(n)^*h_2(n)$，$x_2(n)=0.9^{n-5}R_{20}(n-5)$，$h_2(n)=R_{10}(n)$。

解：在例 6.4 中，已经给出了直接调用 MATLAB 的卷积函数 conv 的方法，也给出了自编卷积计算程序的方法，要注意的是本例时间变量的设定和移位方法。例 6.4 中的时间变量 n 默认地取为 xn 和 hn 的下标，故从 1 开始。而在本例中，设定 n 为从零开始，向量 xn 和 hn 的长度分别为 Nx=20 和 Nh=10；结果向量 y 的长度为 length(y)=Nx+Nh-1。

■ **MATLAB程序q706.m**

```
clear; close all
Nx=20; Nh=10; m=5;          % 设定 Nx, Nh 和位移值 m
n=0:Nx-1;
x1=(0.9).^n;                %产生 x1(n)
x2=zeros(1,Nx+m);
for k=m+1:m+Nx              %产生 x2(n)=x1(n-m)
```

```
    x2(k)=x1(k-m);
end
nh=0:Nh-1; h1=ones(1,Nh);    %产生 h1(n)
h2=h1;                        %h2(n)
y1=conv(x1,h1);  %计算 y1(n)=x1(n)*h1(n)
y2=conv(x2,h2);  %计算 y2(n)=x2(n)*h2(n)
```

程序下段略。

用 stem(n, x1, '.'), stem(nh, h1, '.'), stem(n, y1, '.'), stem(n, x2, '.'), stem(nh, h2, '.'); stem(n, y2, '.') 6 条绘图语句画出 6 个图形。图形分割及标注程序从略。

■ **程序运行结果**

如图 7.6 所示。

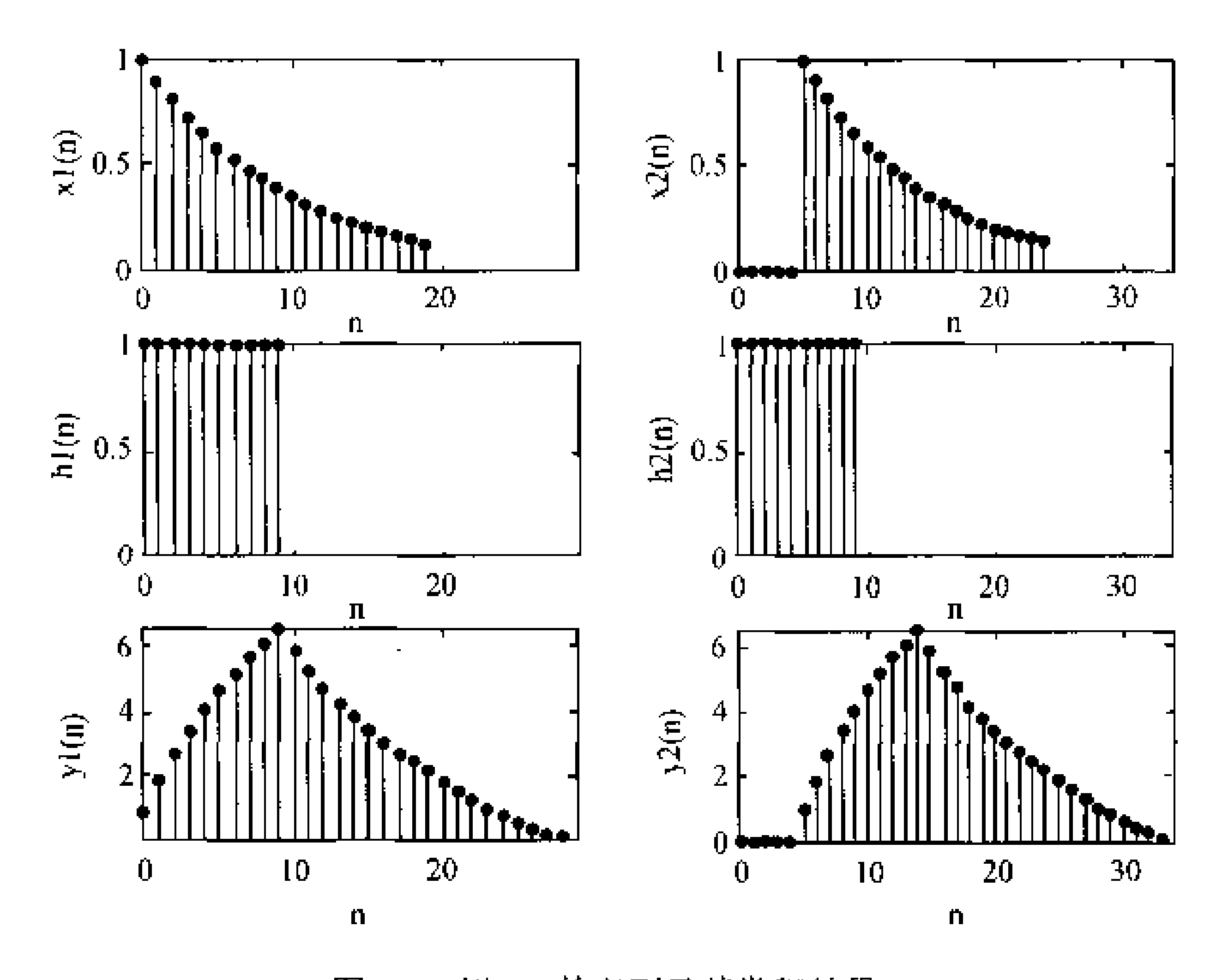


图 7.6　例 7.6 的序列及其卷积结果

7.2　z 变换和傅立叶变换

z 变换是时域离散信号和系统分析及设计的重要数学工具。对于一个序列 $x(n)$，其 z 变换定义为

$$X(z)=\sum_{n=-\infty}^{\infty}x(n)z^{-n}$$

这是个无穷级数，它存在着是否收敛和收敛条件的问题。MATLAB 作数值分析时，是无法求无限长度序列的 z 变换的，这个问题要靠 MATLAB 中的符号运算（Symbolic）工具箱才能解决。

如果 z 变换是 z 的有理分式，虽然其逆 z 变换是无限序列，但求它的系数和指数都是数值计算的范畴，可以用 MATLAB 解决。

如果序列 x 的长度，即 length(x) 有限，其 $n = ns:nf$，则其 z 变换为

$$X(z)=\sum_{n=ns}^{nf}x(n)z^{-n}$$

它是一个 z 的多项式，不存在收敛问题。用 MATLAB 的表达式可写成：

X(z) = x(1) * z^-n(1) + x(2) * z^-n(2) + … + x(end) * z^-n(end)

这是 MATLAB 中信号序列 z 变换的典型形式。它的逆 z 变换一目了然，就是其系数向量 x 和指数向量 n。这也是和连续系统拉氏变换的不同之处。在 s 域，纯粹分子上的 s 多项式属于非物理系统，分母上的 s 的次数必定高于分子。在 z 域，有限长信号序列的 z 变换必定是（纯粹分子上的）z 的多项式，无限长信号序列的 z 变换则是 z 的有理分式，而且其分母上的 z 的次数可以低于分子。

用 z 变换很容易求离散信号 $X(z)$通过线性离散系统 $H(z)$的输出 $Y(z)$：

$$Y(z)=X(z)\,H(z)=\frac{B(z)}{A(z)}$$

它必然是 z 的有理分式

$$Y(z)=\frac{B(z)}{A(z)}=\frac{B(1)+B(2)z^{-1}+\cdots+B(N)z^{-(M-1)}+B(N+1)z^{-M}}{A(1)+A(2)z^{-1}+\cdots+A(N)z^{-(N-1)}+A(N+1)z^{-N}} \tag{7.1}$$

通过长除或逆 z 变换可求出其对应的时域序列。用工具箱函数 residuez 可以求出它的极点留数分解

$$\frac{B(z)}{A(z)}=\frac{r(1)}{1-p(1)z^{-1}}+\frac{r(2)}{1-p(2)z^{-1}}+\cdots+\frac{r(N)}{1-p(N)z^{-1}}+k(1)+k(2)z^{-1}+\cdots$$

其中的 $\boldsymbol{r}$，$\boldsymbol{p}$，$\boldsymbol{k}$ 向量可由下列 MATLAB 语句求得

[r, p, k] = residuez (B, A)

从而得出其逆 z 变换，即时域信号

$$y(n)=r(1)p(1)^n u(n)+r(2)p(2)^n u(n)+\cdots+r(N)p(N)^n u(n)+k(1)\delta(n)+k(2)\delta(n-1)+\cdots$$

其中 k 是当 $M \geqslant N$ 时的直接项，也即有限序列，而其余的则是无限序列。例 7.7 和例 7.8 将说明其应用。

函数 residuez (B, A)要求 A 的首项 A(1)不为零，满足这个条件的有理分式(7. 1)应为

$$Y(z)=z^{-L}\frac{B(z)}{A(z)}=z^{-L}\frac{B(1)+B(2)z^{-1}+\cdots+B(N)z^{-(M-1)}+B(N+1)z^{-M}}{A(1)+A(2)z^{-1}+\cdots+A(N)z^{-(N-1)}+A(N+1)z^{-N}} \tag{7.2}$$

这时先不管 z^{-L}，按留数定理反变换分式部分，再把反变换的结果迟延 L 拍。

实际运用时，很少需要去算 z 变换和逆 z 变换，MATLAB 工具箱提供的 impz 就可以用来求出（7.1）式的逆 z 变换，即其单位脉冲响应。其调用方法为：

➢ [h, T] = impz(B, A, N)

计算 h(n) = IZT[H(z)]。h 为存放 h(n)的列向量，时间变量 N 存放在列向量 T 中。当 N 为标量时，表示 T = [0:N−1]'，计算 h(n)，n = 0，1，2，…，N−1；当 N 为向量时，T=N，仅计算 N 指定的整数点上的 h(n)。

有限长离散序列的傅立叶变换称为离散时间傅立叶变换，它的正逆变换的形式如下：

$$F(\omega)=\sum_{n=-\infty}^{\infty}f(n)\mathrm{e}^{-\mathrm{j}\omega n}$$

$$f(n)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)\mathrm{e}^{\mathrm{j}\omega n}\mathrm{d}\omega$$

可见它在时域上是离散序列，而在频域上是连续函数，即具有连续的频谱。如果时域序列是有限长的，并把它作周期延拓，它的频谱就向一些等间隔的点靠拢。随着延拓周期数的无限增加，其连续频谱就收敛于周期性的离散点，此时就要用离散傅立叶变换来处理。其一般规则如表 7.2 所示。

表 7.2　傅立叶变换一般规则

时域信号（傅立叶反变换）	频谱曲线（傅立叶变换）	变 换 名 称
连续信号	连续频谱	傅立叶变换
离散信号（有限样本点）	周期性连续频谱	离散时间傅立叶变换
多周期离散信号	连续频谱向离散点集中	离散时间傅立叶变换
周期性离散信号	周期性离散频谱	离散傅立叶变换
周期性连续信号	离散频谱	傅立叶级数

MATLAB 信号处理工具箱提供了求连续和离散系统频率响应的两个函数

- freqs　求模拟滤波器 Ha(s)的频率响应函数。

➢ H = freqs(B, A, w)　　计算由向量 w(rad/s)指定的频率点上模拟滤波器 Ha(s)的频率响应 Ha(jw)，结果存于 H 向量中。向量 B 和 A 分别为模拟滤波器系统函数 Ha(s)的分子和分母多项式系数。

[H, w] = freqs(B, A, M)　　计算出 M 个频率点上的频率响应存于 H 向量中，M 个频率存放在向量 w 中。freqs 函数自动将这 M 个频点设置在适当的频率范围。缺省 w 和 M 时 freqs 自动选取 200 个频率点计算。不带左端输出向量时，freqs 函数将自动绘出幅频和相频曲线。

例如，四阶 Butterworth 模拟滤波器归一化低通原型系统函数为

$$H_{\mathrm{a}}(s)=\frac{1}{s^4+2.6131s^3+3.4142s^2+2.6131s+1}$$

用如下程序：

```
B=1; A=[1 2.6131 3.4142 2.6131 1];
w=0: 0.1: 2*pi*5;
freqs (B, A, w)
```

即可绘出其幅频和相频特性曲线，如图 7.7 所示。

- freqz　求数字滤波器 H(z)的频率响应函数。

➢ H=freqz(B, A, w)　　计算由向量 w 指定的数字频率点上数字滤波器 H(z)的频率响应 $H(\mathrm{e}^{\mathrm{j}\omega})$，结果存于 H 向量中。向量 B 和 A 分别为数字滤波器系统函数 H(z)的分子和分母多项式系数。

➢ [H，w]=freqz(B, A, M)　　计算出 M 个频率点上的频率响应存放在 H 向量中，M 个频率存放在向量 w 中。freqz 函数自动将这 M 个频点均匀设置在频率范围[0，π]上。

缺省 w 和 M 时 freqz 自动选取 512 个频率点计算。不带输出向量的 freqz 函数将自动绘出幅频和相频曲线。其他几种调用格式可用命令 help 查阅。

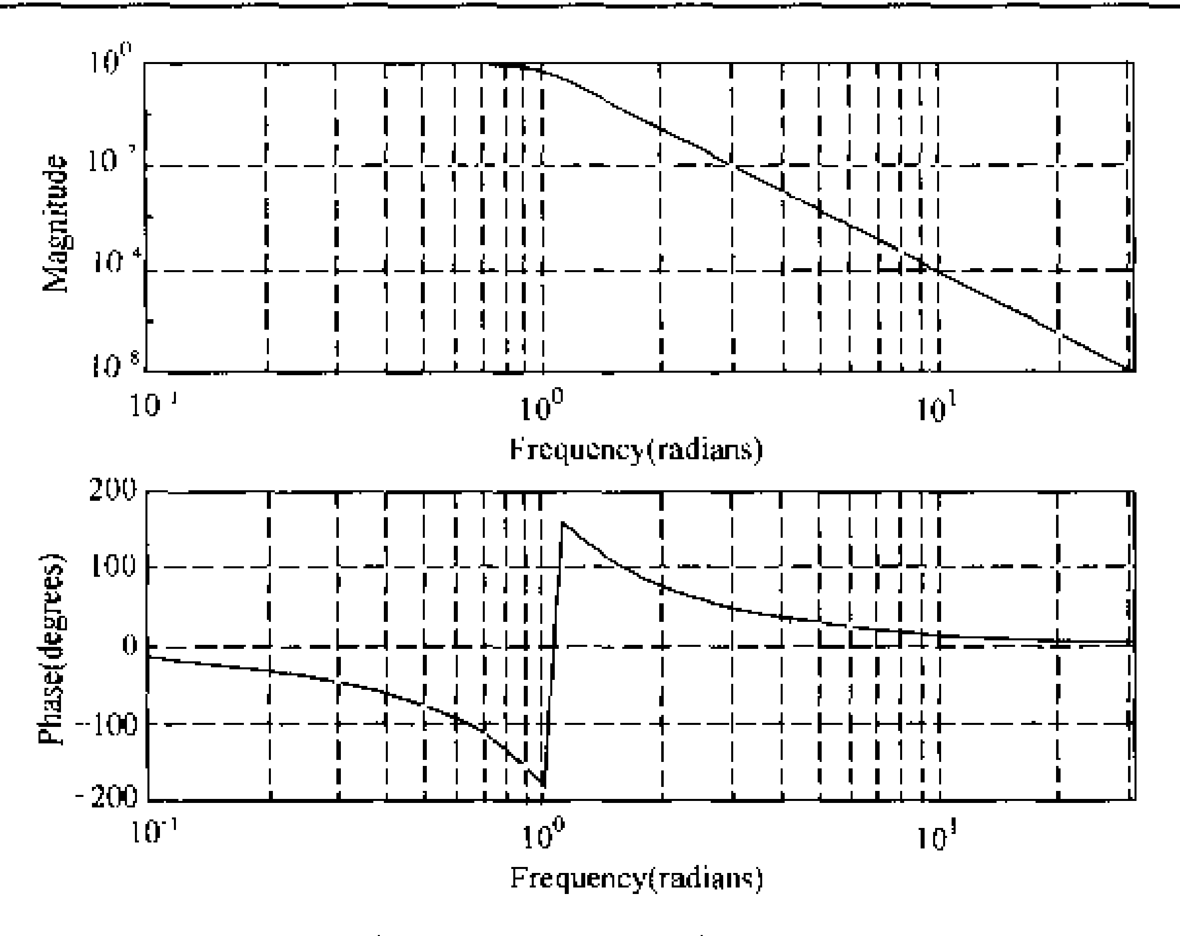


图 7.7 四阶 Butterworth 模拟滤波器的频率响应

例如，八阶梳状滤波器系统函数为

$$H(z) = B(z) = 1 - z^{-8}$$

用下面的简单程序绘出 $H(z)$ 的幅频与相频特性曲线，结果如图 7.8 所示。

```
B = [1 0 0 0 0 0 0 0 -1];
A = 1;
freqz(B, A)
```

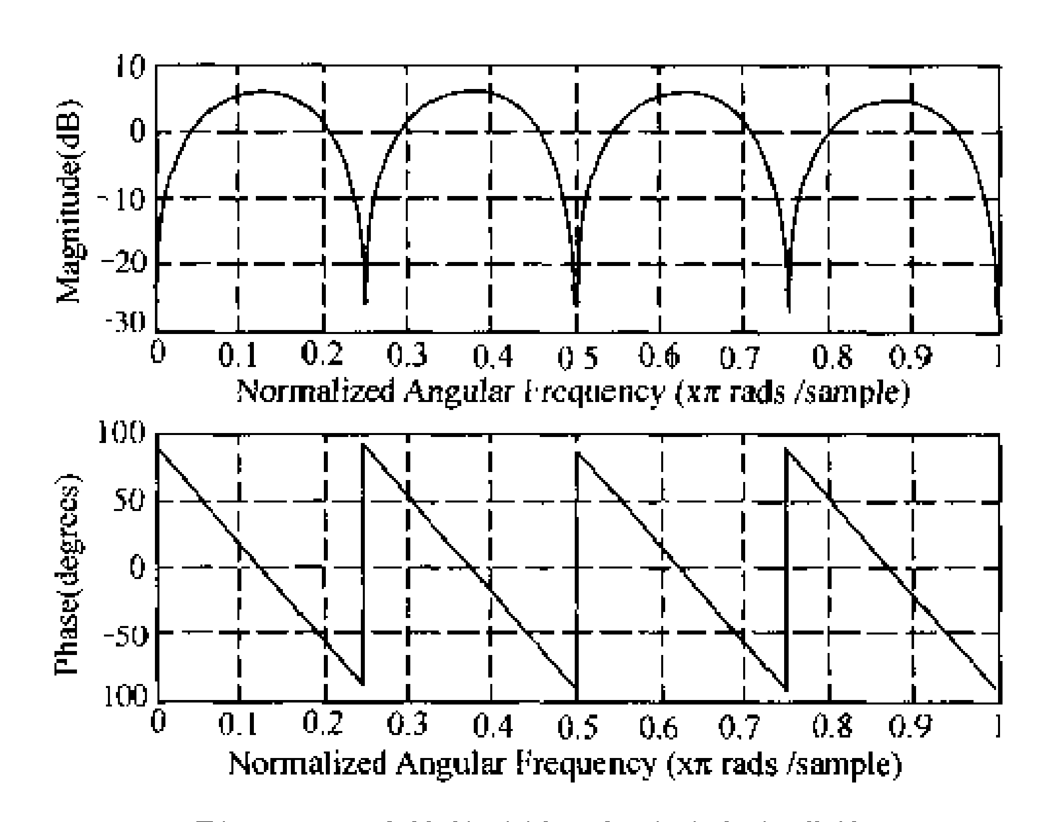


图 7.8 八阶梳状滤波器幅度和相位曲线

采样的逆过程是把采样后的离散序列恢复为原有的连续信号，称为重构。其基本原理是让离散序列通过低通滤波器，所以是一个滤波或卷积计算的问题。其公式为

$$x_a(t) = \sum_{-\infty}^{\infty} x(n) g(t - nT)$$

其特殊之处在于信号 $x(n)$ 是离散的，而脉冲过渡函数 $g(t)$ 是连续的，连续滤波器有两类选择。

（1）物理上可实现的方案，那就是让该信号通过低通滤波器。工程实际中常用的是 D/A 变换加后置平滑滤波器，其中 D/A 变换实际上就是硬件实现的零阶保持器。为了得出连续的输出信号，滤波器采用连续系统的传递函数，也可求出低通滤波器的脉冲响应函数 h 与 x 的卷积 conv(h，x)，不过这时应把 h，特别是输入信号 x 重新表达为时间的连续（即将时间 t 分割得更密）函数。

$$x(t)=\sum_{-\infty}^{\infty}x(n)\delta(t-nT_s)=\cdots+x(-1)\delta(t+T_s)+x(0)\delta(t)+x(1)\delta(t-T_s)+\cdots$$

（2）数学上的仿真，即不考虑因果性限制，允许使用双边脉冲过渡函数的滤波器，例如脉冲过渡函数取 sinc(t)，这时的卷积计算不能用 conv 函数，因为它只适用于有单边脉冲响应的因果系统，所以采用矩阵乘法来完成，详见例 7.11。

【例 7.7】 有限长度序列的 z 变换和逆 z 变换

已知两序列 $x_1=[1，2，3]$，$n_1=[-1:1]$ 及 $x_2=[2，4，3，5]$，$n_2=[-2:1]$，求出 x_1 与 x_2 及其卷积 x 的 z 变换。

解：■ 建模

给定有限离散序列 x 及其时间变量向量 $\boldsymbol{n}=[ns:nf]$，其中 ns 和 nf 分别表示 x 的时间变量的起点和终点，则其 z 变换可写成

$$X(z)=\sum_{n=-\infty}^{\infty}x(n)z^{-n}=\sum_{n=ns}^{nf}x(n)z^{-n}$$

用 MATLAB 的表达式可写成

X=x(1)*z^-n(1)+x(2)*z^-n(2)+ …+x(end)*z^-n(end)

因此，本例中 x_1 和 x_2 的 z 变换为

$$X_1(z)=z+2+3z^{-1}，\quad X_2(z)=2z^2+4z+3+5z^{-1}$$

根据 z 变换的时域卷积定理，只要求出 $X(z)=X_1(z)X_2(z)$ 即可。这是两个多项式相乘，可用 conv 函数来求得序列 x(n)，x(n)应该也是 x 和 n 两个数组。conv 函数只能给出 x 数组，n 数组要自己判别。n 的起点 ns＝ns1 ＋ ns2＝−3，终点 nf＝nf1 ＋ nf2＝2。由 x 和 n 即可得出 X(z)。因此可得计算 x(n)的程序如下。

■ MATLAB程序q707.m

```
%计算 z 变换
x1=[1, 2, 3];          ns1=-1;          % 设定 x1 和 ns1
nf1=ns1+length(x1)-1;                   % nf1 可以算出
x2=[2, 4, 3, 5];ns2=-2;                 % 设定 x2 和 ns2
nf2=ns1+length(x1)-1;                   % nf2 可以算出
x=conv(x1, x2)                          % 求出 x
n=(ns1+ns2): (nf1+nf2)                  % 求出 n
```

■ 程序运行结果

```
x =
     2     8    17    23    19    15
n =
    -3    -2    -1     0     1     2
```

由此可得

$$X(z)=2z^3+8z^2+17z+23+19z^{-1}+15z^{-2}$$

$X(z)$是 z 的多项式形式，它的逆变换就是其系数组成的向量 $\boldsymbol{x}$ 和 z 的幂次组成的时间向量 $\boldsymbol{n}$，所以是有限长度序列。

建议读者将程序 q707.m 写成一个通用卷积函数 conv_m.m，以便直接调用，其首行格式为

➢ function [y，ny]=conv_m(x，nsx，h，nsh)

【例 7.8】 求 z 多项式分式的逆变换。

设系统函数为 $W(z)=\dfrac{-3z^{-1}}{2-2.2z^{-1}+0.5z^{-2}}$，输入例 7.7 中的 x_2 信号，用 z 变换计算输出 $y(n)$，$y(n)=\text{IZT}[Y(z)]$。

解：■ 建模

由例 7.7 可知 $X_2(z)=2z^2+4z+3+5z^{-1}$

故　　$Y(z)=X(z)W(z)=z^{-nsy}\dfrac{B(z)}{A(z)}$

其中：$B=\text{conv}(-3,[2,4,3,5])$，$A=[2,-2.2,0.5]$，$nsy=$ 分母分子多项式 z 的最高幂次之差。

调用 [r, p, k] = residuez(B, A)，可由 B，A 求出 r，p，k，进而求逆 z 变换，得

$$y(n)=r(1)p(1)^{n-nsy}u(n-nsy)+r(2)p(2)^{n-nsy}u(n-nsy)+k(1)\delta(n-nsy)+k(2)\delta(n-nsy-1)$$

其中 $u(n-n_0)$ 和 $\delta(n-n_0)$分别为在 n_0 处的单位阶跃函数及单位脉冲函数。

■ MATLAB程序q708.m

```
clear, close all
x=[2,4,3,5];nsx=-2;            % 输入序列及初始时间
nfx=nsx+length(x)-1;           % 计算序列终止时间
Bw=-3;  nsbw=-1;               % 系统函数的分子系数,及 z 的最高次数
Aw=[2,-2.2,.5];  nsaw=0; % 系统函数的分母系数,及 z 的最高次数
B=conv(-3,x);                  % 输入与分子 z 变换的多项式乘积
A=Aw;                          % 分母不变
nsy=nsaw-(nsbw-nsx)            % 分子比分母 z 的次数高 nsy
[r,p,k]=residuez(B,A)          % 求留数 r,极点 p 及直接项 k
nf=input('终点时间 nf= ');     % 要求键入终点时间
n = min(nsx,nsy):nf;           % 生成总时间数组
% 求无限序列 yi 和直接序列 yd
yi=(r(1)*p(1).^(n-nsy)+r(2) *p(2).^(n-nsy)).*stepseq(nsy,n(1),nf);
yd =k(1) *impseq(nsy,n(1),nf)+k(2) *impseq(-1-nsy,n(1),nf);
y=yi+yd;                                          % 合成输出
xe = zeros(1,length(n));                          % 初始化,将 x 延拓为 xe,
xe(find((n>=nsx)&(n<=nfx))==1)=x;                 % 在对应的 n 处把 xe 赋值 x
subplot(2,1,1),stem(n,xe,'.'),line([min(n(1),0),nf],[0,0]) % 绘图
subplot(2,1,2),stem(n,y,'.'),line([min(n(1),0),nf],[0,0])
subplot(2,1,1),stem(n,xe,'.'),
line([min(n(1),0),nf],[0,0])    % 绘图
subplot(2,1,2),stem(n,y,'.'),
line([min(n(1),0),nf],[0,0])
```

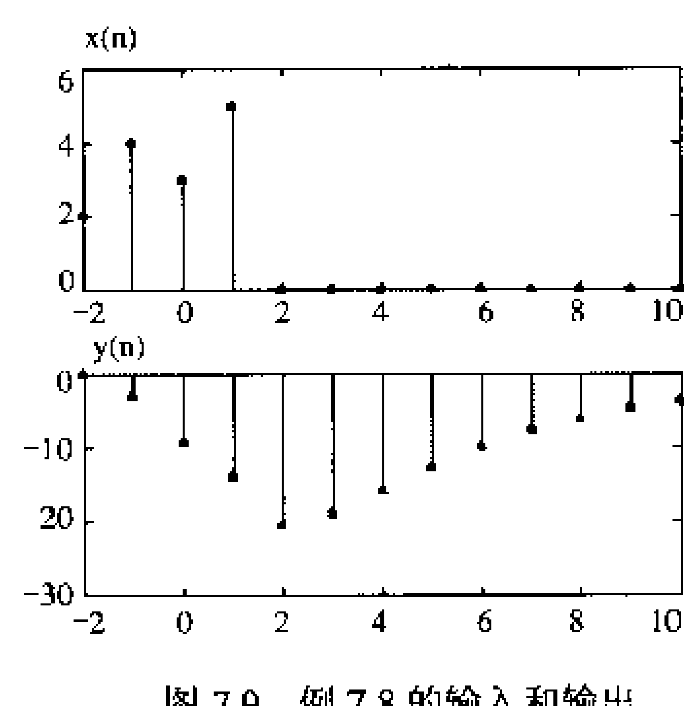


图 7.9 例 7.8 的输入和输出

■ **程序运行结果**

```
nsy =-1
r =  -57.7581
     204.7581
p =     0.7791
        0.3209
k =  -150   -30
```

由（7.8）式写出 $Y(z)$的部分分式如下：

$$Y(z)=z\left(\frac{r(1)}{1-p(1)z^{-1}}+\frac{r(2)}{1-p(2)z^{-1}}+k(1)+k(2)z^{-1}\right)$$

所以，由(7.9)式得到 $y(n)$表达式：

$$y(n)=-57.7581\times0.7791^{n+1}u(n+1)+204.7581\times0.3209^{n+1}u(n+1)-150\delta(n+1)-30\delta(n)$$

$x(n)$和 $y(n)$的图形见图 7.9。

如果不需要求出表达式，那本题就可以直接用工具箱函数 filter 来解，键入

```
x=[2,4,3,5,zeros(1, 10)]; Bw=-3; Aw=[2,-2.2,.5];
y=filter(Bw,Aw,x), stem(y)
```

所得 y 的波形与图 7.9 相同，只不过 x 和 y 都从 n=1 开始，而长度则取决于输入 x 的长度，故在 x 给定序列的后面加了 10 个 0。

【例 7.9】 离散时间傅立叶变换

取一个周期的正弦信号，作 8 点采样，求它的连续频谱。然后对该信号进行 N 个周期延拓，再求它的连续频谱。把 N 无限增大，比较分析其结果。

解：■ 建模

为了求离散信号的连续频谱，不能直接用 fft 函数，可从离散时间傅立叶变换的定义出发。

$$X(\omega)=\sum_{n=-\infty}^{\infty}x(n)\mathrm{e}^{-\mathrm{j}\omega n}$$

■ **MATLAB实现**

设置一系列较密频率 wi，就可求出一系列 X(wi)=x*exp(-j*wi*n')，因为 x 是行向量，n' 以及 exp(-j*wi*n')为同长的列向量，两者的向量点乘就包含了逐项相乘后的连加演算，得出一个标量 X(wi)，即该频点上的频率响应。不同的 wi 可用 for 循环来解决。但 MATLAB 中往往可以用元素群运算来代替一个 for 循环，为此把 w 也设成行向量，放在 n'之后，n'*w 及 exp(-j*n'*w)就成为一个矩阵，其行数与 x 相同，列数与 w 的长度相同。用 x 左乘它以后，各列分别得出相应 wi 处的响应，最后得到 X = x*exp(-j*w*n')。

可以把离散时间傅立叶变换写成一个子程序 dtft.m：

```
function X=dtft(x, w)
% x 为输入离散序列，时间数组为增序整数，故可用下标表示
% w 为频率数组，由于 exp(-j*n'*w) 以 2π为周期，
% 所以其范围通常选为[-π, π]或[0, 2π]
X = x*exp(-j*[1:length(x)]'*w);
```

注意　在此定义下的时间变量和频率变量都是无量纲的，要恢复其原来量纲，应把无量纲时间 n 乘以采样周期 Ts，而把无量纲频率 w 乘以采样频率 Fs。

■ MATLAB程序q709.m

```
disp('八点时间信号的离散时间傅立叶变换')
x0=sin(2*pi* [1:8]/8) *5;                    % x0是8点行向量
dt=2*pi/8;
w=linspace(0,2*pi,1000)/dt;                  % w是1000点行向量
X0 = dtft(x0,w) *dt;                         % 求得频率响应X0
subplot(3,1,1),plot(w,abs(X0)),grid,shg       % 画图
disp('重复N次的八点时间信号的离散时间傅立叶变换')
N=input('N= ');                              % 用键盘输入延拓周期数
x1=reshape(x0'*ones(1,N),1,N*length(x0));     % 延拓后的时域信号x1
X1 = dtft(x1,w) *dt;                         % 求x1的频率响应X1
subplot(3,1,2),plot(w,abs(X1)),grid,shg       % 画图
disp('重复无穷次的八点信号的离散傅立叶变换')
pause,X2=fft(x0*dt);                          % 离散傅立叶变换
w1=2*pi* [0:length(x0)-1]/length(x0);         % 离散频点向量
subplot(3,1,3),stem([-w1,w1],[abs(X2),abs(X2)]),grid,shg
axis([min(w),max(w),0,max(abs(X2))]),grid,shg
% 标注语句从略
```

■ 程序运行结果

执行程序并按提示键入N = 4，所得图形如图7.10所示。N取得愈大，其峰值愈大，宽度愈窄。当N取得很大时，会出现内存不足的问题，这是用矩阵乘法做傅立叶变换的缺点。另外，因为那时峰值点处的宽度很窄，也会出现所选频点对不上峰值点的问题。所以对于N无限增大的情况，必须用fft函数来求。这时用连续频谱也没有意义了。这里用同样的横坐标把几种频谱进行对比，使读者更好地理解其关系。

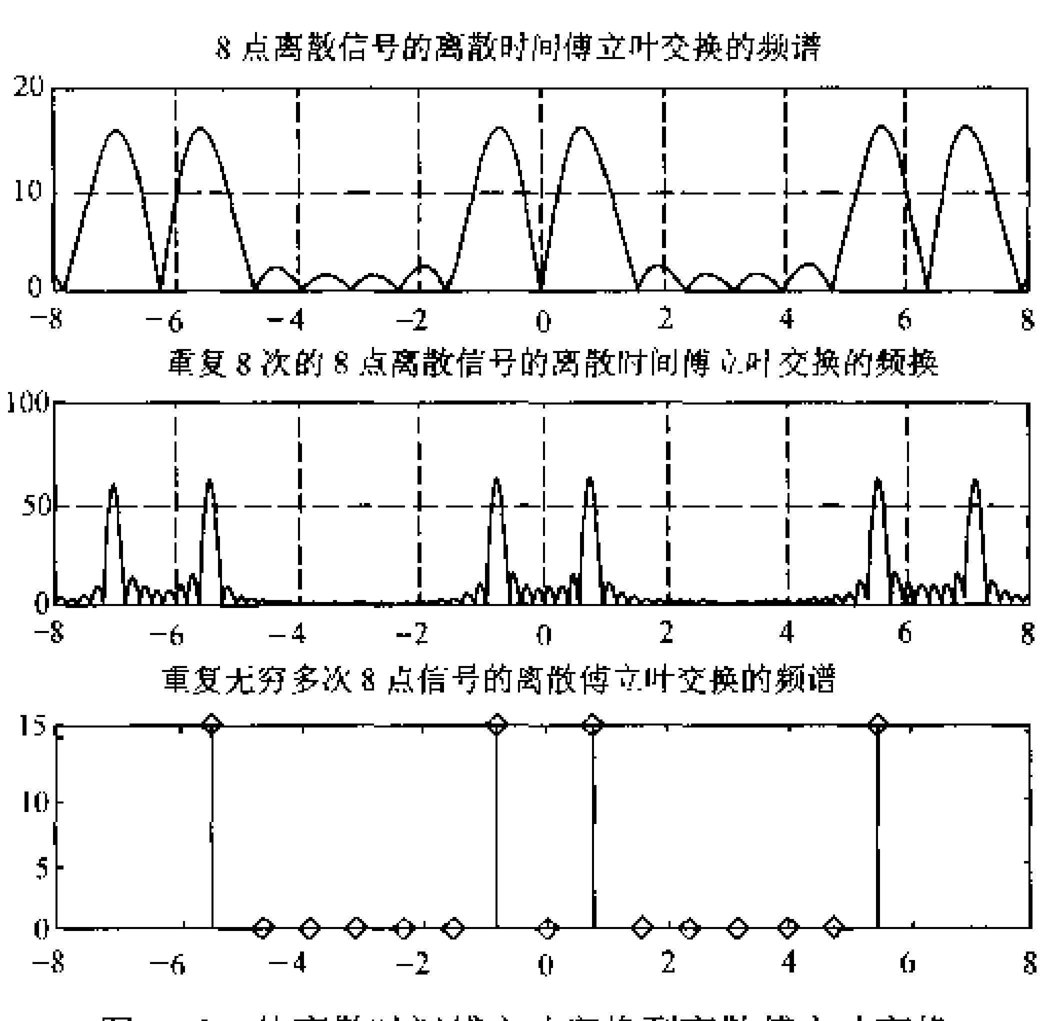


图7.10　从离散时间傅立叶变换到离散傅立叶变换

【例 7.10】 时域采样频率与频谱混叠

$$x_a(t)=Ae^{-\alpha t}\sin\Omega_0 t u(t),\ A=444.128,\quad \alpha=50\sqrt{2}\pi,\quad \Omega_0=50\sqrt{2}\pi。$$

计算并图示 $x_a(t)$ 及其幅频特性函数 $|X_a(j\Omega)|$。

分别以采样频率 $f_s=1000$ Hz，400Hz 和 200Hz 对 $x_a(t)$ 进行等间隔采样，得到 $x(n)=x_a(nT)$，$T=1/f_s$ 为采样周期。计算并图示三种采样频率下的采样信号 $x(n)$ 及其幅频特性函数 $|X(e^{j\omega})|_{\omega=\Omega T}=\left|X(e^{j\Omega T})\right|$。观察 $X(e^{j\Omega T})$ 的周期性、周期以及频谱混叠程度与 f_s 的关系。

解：■ 建模

对 $x_a(t)$ 进行等间隔采样，得到 $x(n)=x_a(nT)$，$T=1/f_s$ 为采样周期。如果 $X_a(j\Omega)$=FT[$x_a(t)$]，则

$$X(e^{j\Omega T})=\mathrm{FT}[x(n)]=\sum_{n=-\infty}^{\infty}x(n)e^{j\Omega nT}=\frac{1}{T}\sum_{k=-\infty}^{\infty}X_a\left(j\left(\Omega-\frac{2\pi k}{T}\right)\right)$$

由以上关系式可见，采样信号的频谱函数是原模拟信号频谱函数的周期延拓，延拓周期为 $2\pi/T$。如果以频率 f 为自变量（$\Omega=2\pi f$），则以采样频率 $f_s=1/T$ 为延拓周期。对频带限于 f_c 的模拟信号 $x_a(t)$，只有当 $f_s\geqslant 2f_c$ 时，采样后 $X(e^{j\Omega T})$ 才不会发生频谱混叠失真。这就是著名的采样定理。

严格地讲，MATLAB 无法计算连续函数 $X_a(j\Omega)$。但工程上可认为，当 f_s 足够大时，频谱混叠可忽略不计，从而可对采样序列进行傅立叶变换，得到 $X_a(j\Omega)$。

程序分别设定 4 种采样频率 f_s=10kHz，1kHz，400Hz 和 200Hz，对 $x_a(t)$ 进行采样，得到采样序列 $x_a(t)$，$x_{a1}(n)$，$x_{a2}(n)$，$x_{a3}(n)$，画出其幅度频谱。采样时间区间均为 0.1 秒。为了便于比较，画出了幅度归一化的幅频曲线，如图 7.11 所示。

■ MATLAB程序q710.m

```
%时域采样及其频谱
clear;close all;
fs=10000; fs1=1000; fs2=400; fs3=200; % 设置四种采样频率
t=0:1/fs:0.1;                          % 采集信号长度为 0.1 秒
A=444.128; a=50*sqrt(2)*pi; b=a;
xa=exp(-a*t).*sin(b*t);
k=0:511; f=fs*k/512;                   % 由 wk=2πk/512=2πfT 求得模拟频率 f
Xa=dtft(xa,2*pi*k/512);                % 近似模拟信号频谱
T1=1/fs1;t1=0:T1:0.1;                  % 采集信号长度为 0.1 秒
x1=A*exp(-a.*t1).*sin(b*t1);           % 1kHz 采样序列 x1(n)
X1=dtft(x1,2*pi*k/512);                % x1(n)的 512 点 dtft
T2=1/fs2;t2=0:T2:0.1;                  % 采集信号长度为 0.1 秒
x2=A*exp(-a.*t2).*sin(b.*t2);          % 400Hz 采样序列 x2(n)
X2=dtft(x2,2*pi*k/512);                % x2(n)的 512 点 dtft
T3=1/fs3;t3=0:T3:0.1;                  % 采集信号长度为 0.1 秒
x3=A*exp(-a.*t3).*sin(b.*t3);          % 200Hz 采样序列 x3(n)
X3=dtft(x3,2*pi*k/512);                % x3(n)的 512 点 dtft
figure(1);
```

```
subplot(3,2,1);plot(t,xa);              % 画出原始波形
axis([0,max(t),min(xa),max(xa)]);title('模拟信号');
xlabel('t (s)');ylabel('Xa(t)'); line([0,max(t)],[0,0])
subplot(3,2,2);plot(f,abs(Xa)/max(abs(Xa)));
title('模拟信号的幅度频谱');axis([0,500,0,1])
xlabel('f (Hz)');ylabel('|Xa(jf)|');
……
用 stem(n1,x1,'.'); plot(f,abs(X1)/max(abs(X1)));
 stem(n2,x2,'.'); plot(f,abs(X2)/max(abs(X2)));
和 stem(n3,x3,'.'); plot(f,abs(X3)/max(abs(X3)));
```

语句画出其余三种情况的曲线。图形划分及标注语句从略。

■ 程序运行结果见图 7.11

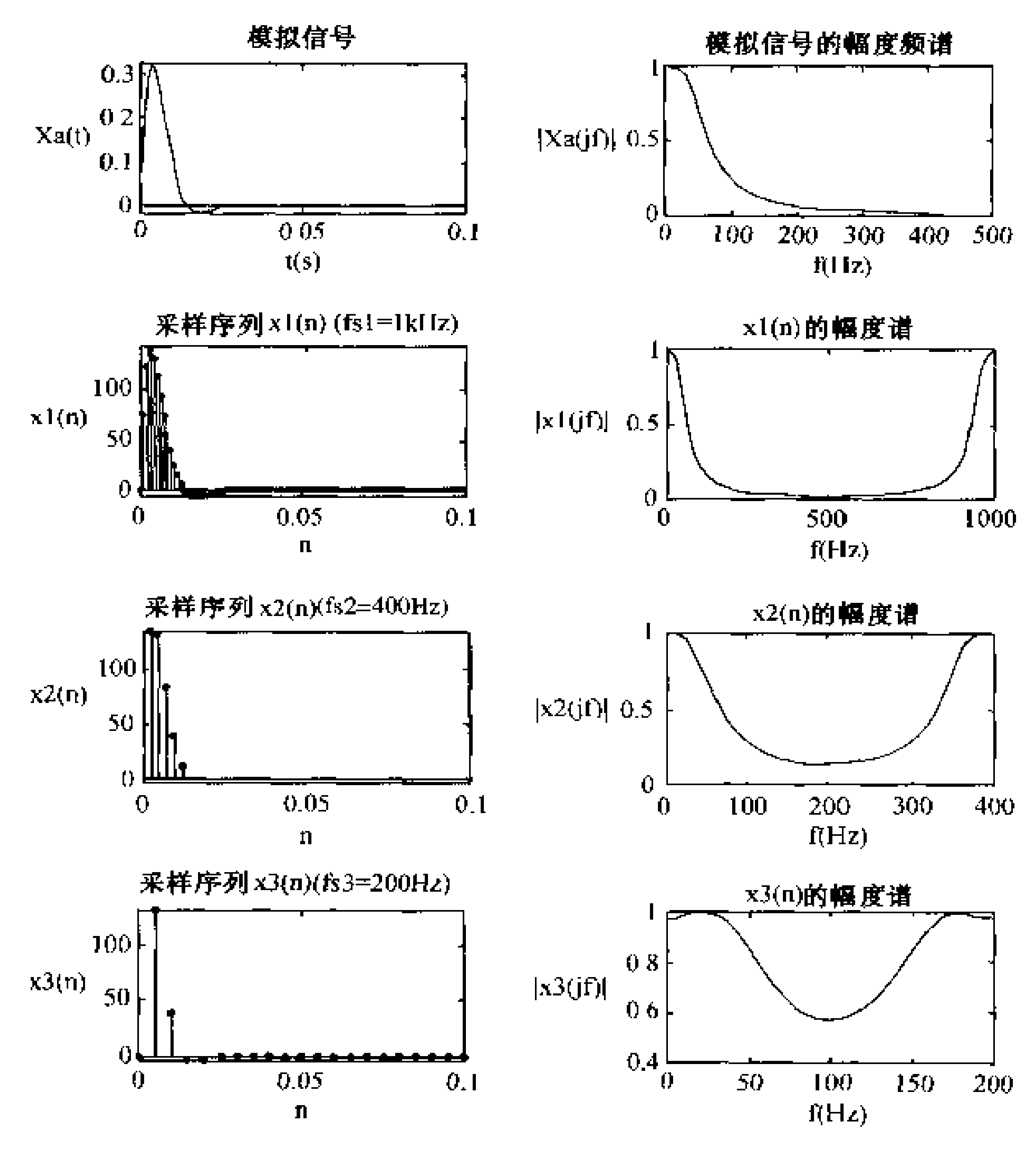


图 7.11 模拟信号的采样及其频谱

由图 7.11 可见，当 f≥500Hz 时，|Xa(jΩ)|的值很小。所以，fs = 1kHz 的采样序列 xa1(n) 的频谱混叠很小；而 fs = 400Hz 时，频谱混叠较大；fs = 200Hz 时，频谱混叠最严重。图 7.11 中，X1(jf)，X2(jf)，X3(jf) 均以相应的采样频率（1kHz，400Hz，200Hz）为周期。

以乃奎斯特频率 fs/2 处的频谱幅度来比较其混叠，此处 k = 256，键入以下语句，得到：

```
k=256;
```

```
abs(Xa(k))/max(abs(Xa))
ans =  2.4651e-004
abs(X1(k))/max(abs(X1))
ans =     0.0247
abs(X2(k))/max(abs(X2))
ans =     0.1534
abs(X3(k))/max(abs(X3))
ans =     0.5703
```

可以看出，随着采样频率的减小，混叠现象加大。

【例 7.11】 由离散序列恢复模拟信号

用时域内插公式 $$x_a(t)=\sum_{n=-\infty}^{\infty}x(n)g(t-nT)$$

其中 $$g(t)=\frac{\sin(\frac{\pi}{T}t)}{\frac{\pi}{T}t}=\text{sinc}(F_s t)$$

模拟用理想低通滤波器恢复 $x_a(t)$ 的过程，观察恢复波形，计算出最大恢复误差。其中 $x_a(t)$ 和 $x(n)$ 同例 7.10。采样频率 F_s 取 400 Hz 及 1000Hz 两种作比较。

解：■ 建模

所谓模拟信号恢复（或重构）就是根据离散点的采样序列 $x(n)=x_a(nT)$ 估算出采样点之间的模拟信号的值。因此，$g(t)$应是一个连续时间函数。MATLAB 不能产生连续函数。但可以把 t 数组取得足够密，使在一个采样周期 T 中，插入 m 个点，也即使 $dt=T/m$，就可以近似地将 $g(t)=\text{sinc}(t/T)$看做连续波形。

根据上述内插公式，在用 MATLAB 实现时，设定一个 ti 值求 xa(ti)的问题，可归结为一个行向量 x(n)和一个同长的由 n'构成的列向量 g(ti−n'T)相乘，即 xa(ti) = x*g(ti)，这里面已包括了求和运算，和例 7.10 求频谱的算法非常相似。对于很多个 ti，既可以用 for 循环，也可把 t 作为行向量代入，利用 MATLAB 元素群运算的规则，一次求出全部的 xa(t)。在 t−n'T 中，t 设成行向量，n'T 为列向量。我们的目的是把它构成一个行数与 n 同长而列数与 t 同长的矩阵，因此，要把两项分别扩展为这样的矩阵。这只要把 t 右乘列向量 ones(length(n), 1)，把 n'T 左乘行向量 ones(1，length(t))即可。

所以，只要正确设定 t 向量和 n 向量，设 t 向量长 M， n=1:N−1，就可生成 t−n'T 矩阵，把它命名为 TNM，用 MATLAB 语句表示为：

➢ TNM=ones(length(n)，1) t− n'*T*ones(1，length(t))

其运算结果为如下矩阵

$$\text{TNM}=\begin{bmatrix} t(1) & t(2) & \cdots & t(M) \\ t(1)-T & t(2)-T & \cdots & t(M)-T \\ t(1)-2T & t(2)-2T & \cdots & t(M)-2T \\ \vdots & \vdots & \ddots & \vdots \\ t(1)-(N-1)T & t(2)-(N-1)T & \cdots & t(M)-(N-1)T \end{bmatrix}$$

因此，MATLAB 中内插公式可简化为

xa = x * g (TNM) = x * G

用 sinc 函数内插时

G = sinc(Fs * TNM)

G 是一个与 TNM 同阶的矩阵。N 为序列 x(n)的长度，M 为 t 的点数，通常有

M = (N−1) * m +1

由此可编写本例中用 xa1(n)和 xa2(n)重构 xa(t)的程序。

■ MATLAB程序q711.m

```
% 时域采样与重构
clear; close all;
A=444.128; a=50*sqrt(2)*pi; b=a;
for k=1:2
   if k==1 Fs=400;
   elseif k==2 Fs=1000; end
   T=1/Fs;    dt=T/3;                      %每个采样间隔 T 上 g(t)取三个样点
   Tp=0.03;                                % 重构时间区间为[0,0.03s]
   t=0:dt:Tp;                              % 生成序列 t
   n=0:Tp/T;                               % 生成序列 n
   TMN=ones(length(n),1)*t- n'*T*ones(1,length(t)); % 生成 TNM 矩阵
   x=A*exp(-a*n*T).*sin(b*n*T);            % 生成模拟信号采样序列 x(n)
   xa=x*sinc(Fs*TMN);                      % 内插公式
   subplot(2,1,k);plot(t,xa);hold on
   axis([0,max(t),min(xa)-10,max(xa)+10]);
   st1=sprintf('由 Fs= %d',Fs);            % 生成标注字符串左端,含变动部分 Fs
   st2='Hz 的采样序列 x(n)重构的信号';       % 生成标注字符串右端
   st=[st1,st2]; title(st)                 % 拼装成一个字符串并显示在标题上
   ylabel('xa(t)');
   xo=A*exp(-a*t).*sin(b*t);               % 以插值频率对原始模拟信号采样
   stem(t,xo,'.');line([0,max(t)],[0,0])
   emax2=max(abs(xa-xo))                   % 求插值结果与原数据的差
end
```

在本程序的编写中，用 for 循环来处理两种不同采样频率的计算，减少了许多重复的语句。读者可特别注意如何使两个图的标题在两次循环中改变。

■ 程序运行结果

输出最大重构误差如下

```
emax2 =    27.7015
emax1 =     9.9436
```

内插结果如图 7.12 中的连续曲线所示，图中的离散序列是原始模拟信号 xa(t) 的采样真值。从图 7.12 和最大重构误差（emax2，emax1）中容易看出，Fs = 1000Hz 时的采样序列 xa1(n) 内插重构的信号误差比 Fs = 400Hz 时小得多。可见，误差主要由频率混叠失真引起。当然，采样序列 x(n)的样本数较少也会使误差增大。另外，xa(t) 变化愈缓慢处误差愈小。

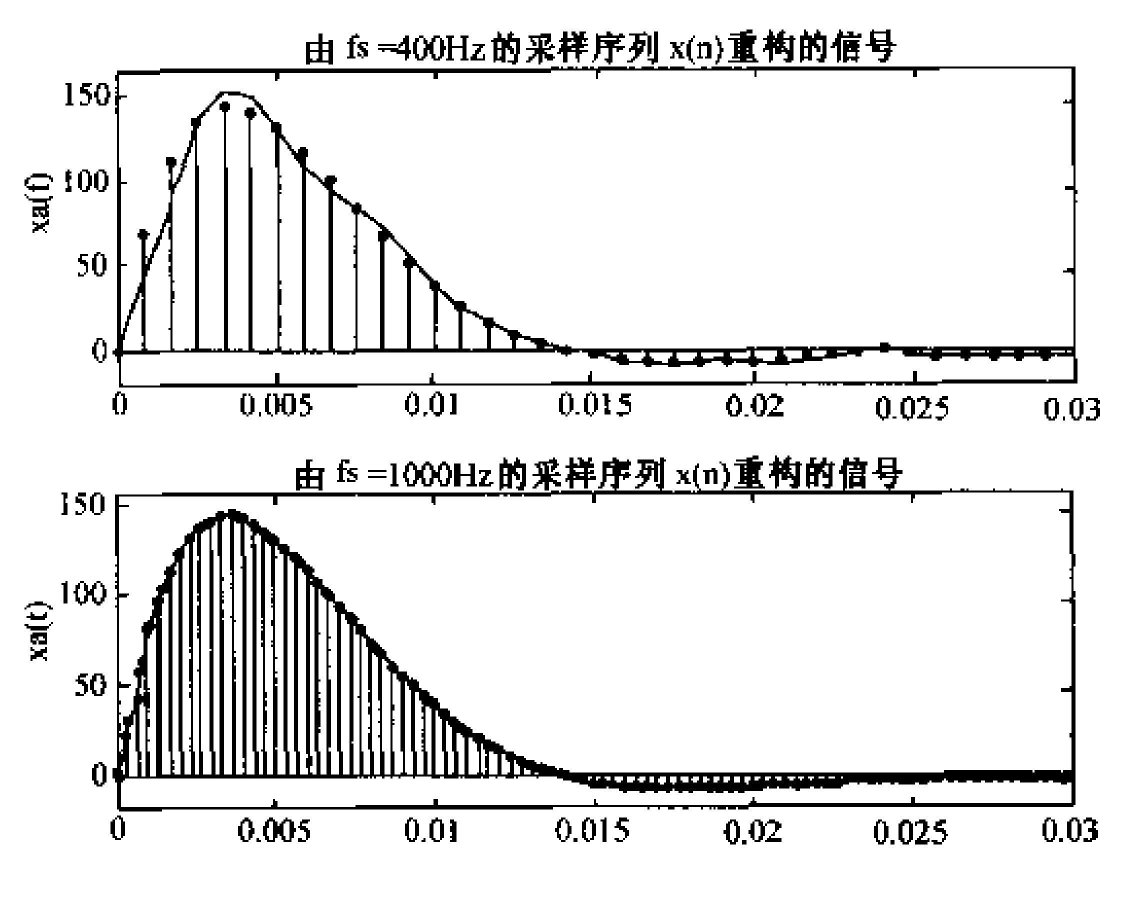


图 7.12　用 sinc 函数内插重构信号波形

由于内插函数 g(t)的采样间隔 dt 为 x(n)的采样间隔 T 的三分之一（即 T = 3dt），所以，不难验证，误差数组 xa-xo 每隔两点就出现一次零。即在这些点上，离散序列原有值 xo 等于插值序列 xa 的值，与时域内插理论相吻合。

【例 7.12】 梳状滤波器零极点和幅频特性

梳状滤波器系统函数有如下两种类型。

FIR 型：$H_1(z)=1-z^{-N}$

IIR 型：$H_2(z)=\dfrac{1-z^{-N}}{1-a^N z^{-N}}$

分别对 $N = 8$，$a = 0.8$，0.9，0.98 计算并图示 $H_1(z)$ 和 $H_2(z)$ 的零点、极点及幅频特性曲线。评述各个曲线，说明 FIR 系统和 IIR 系统的特点以及极点位置的影响。

在求解本题之前先简单说明一下题中将用到的绘图函数。

- zplane　H(z)的零-极点图绘制。

➢ zplane(z，p)　绘制出列向量 z 中的零点(以符号“o”表示)和列向量 p 中的极点(以符号“×”表示)以及参考单位圆。并在多阶零点和极点的右上角标出其阶数。如果 z 和 p 为矩阵，则 zplane 以不同的颜色分别绘出 z 和 p 各列中的零点和极点。

➢ zplane(B，A)　绘出系统函数 H(z) 的零点和极点图，其中 B 和 A 为 H(z) = B(z)/A(z) 的分子和分母多项式系数向量。

解：调用函数 freqz 和 zplane 很容易写出程序 q712.m。程序运行结果如图 7.13 所示。由图可以看出，阶数相同时，IIR 梳状滤波器具有更平坦的通带特性和更窄的过渡带，极点距单位圆越近，这一特性就越明显。

■ **MATLAB程序q712.m**

```
% 梳状滤波器幅频特性与极点位置
clear;close all
b=[1,0,0,0,0,0,0,0,-1];            % H1(z)和H2(z)的分子多项式系数向量
a0=1;                              % H1(z)的分母多项式系数向量
```

```
a1=[1, 0, 0, 0, 0, 0, 0, 0, -(0.8)^8];        % H2(z)的分母多项式系数向量(a=0.8)
a2=[1, 0, 0, 0, 0, 0, 0, 0, -(0.9)^8];        % H2(z)的分母多项式系数向量(a=0.9)
a3=[1, 0, 0, 0, 0, 0, 0, 0, -(0.98)^8];       % H2(z)的分母多项式系数向量(a=0.98)
[H, w]=freqz(b, a0);                          %H1(z)的频响函数
[H1, w1]=freqz(b, a1);                        %H2(z)的频响函数(a=0.8)
[H2, w2]=freqz(b, a2);                        %H2(z)的频响函数(a=0.9)
[H3, w3]=freqz(b, a3);                        %H2(z)的频响函数(a=0.98)
...
```

以下为绘图语句，用 zplane(b, a0)，zplane(b, a1)，zplane(b, a2)，zplane(ba3)四条语句分别画出四种情况的零极点，用 plot(w/pi，abs(H))，plot(w1/pi，abs(H1))，plot(w2/pi，abs(H2))，plot(w3/pi, abs(H3)) 四条语句分别画出四种情况的幅频特性，标注语句从略。

■ **程序运行结果见图 7.13**

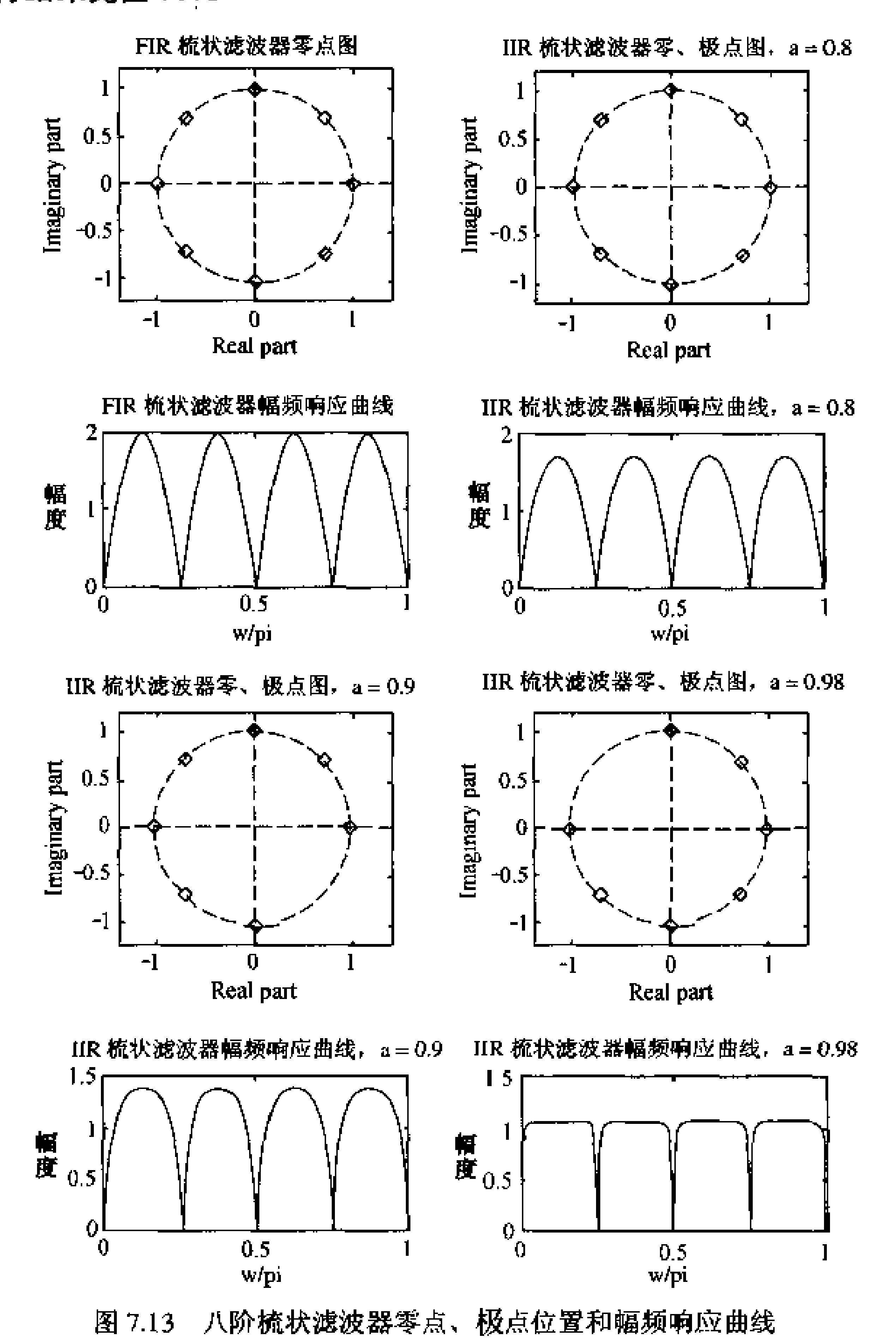


图 7.13　八阶梳状滤波器零点、极点位置和幅频响应曲线

【例 7.13】 低通滤波效果及傅立叶变换时域卷积定理验证

设低通数字滤波器系统函数为

$$H(z)=\frac{0.0003738(1+z^{-1})^6}{(1-1.2686z^{-1}+0.7051z^{-2})(1-1.0106z^{-1}+0.3583z^{-2})(1-0.9044z^{-1}+0.2155z^{-2})}$$

产生输入信号 $x(n)=\cos(0.04\pi n)+\cos(0.08\pi n)+\cos(0.4\pi n)+0.3\omega(n)$，$0\leqslant n\leqslant 63$

其中 $\omega(n)$ 是均值为 0，方差为 1 的白噪声序列。

计算滤波器对 $x(n)$ 的响应输出 $y(n)$，并图示 $x(n)$ 和 $y(n)$，观察滤波效果。

计算和图示 $|H(e^{j\omega})\cdot X(e^{j\omega})|$，并与 $|Y(e^{j\omega})|$ 比较，用傅立叶变换的时域卷积定理加以解释。

解：■ 建模

如前所述，只要求出 $H(z)=B(z)/A(z)$ 的分子和分母多项式系数向量 $\boldsymbol{B}$ 和 $\boldsymbol{A}$，则可调用滤波器直接 II 型实现函数 filter 对输入信号 $x(n)$ 进行滤波。

➢ y = filter(B, A, x)

调用多项式相乘函数 conv 可求得本题 H(z) 的分子和分母多项式系数向量 B 和 A。

调用函数 freqz 可求出 $|H(e^{j\omega})|$。

调用函数 fft 可求出 $|X(e^{j\omega})|$ 和 $|Y(e^{j\omega})|$。

■ MATLAB 程序 q713.m

```
%IIR 滤波器实现及 FT 的时域卷积定理验证
clear; close all
%产生输入信号 x(n) 向量 x
n=0:255;  N=4096;
x=cos(0.04*pi*n)+cos(0.08*pi*n)+cos(0.4*pi*n);
w=randn(size(x));                %产生正态零均值噪声
x=x+0.3*w;
%求 H(z) 分子分母多项式系数向量 B 和 A
b=[1,2,1];                       %(1+z^-1)^2 的展开系数
%嵌套调用卷积函数 conv,计算 (1+z^-1)^6 的展开系数向量 B
B=0.0003738*conv(conv(b,b),b);
a1=[1,-1.2686,0.7051];
a2=[1,-1.0106,0.3583];
a3=[1,-0.9044,0.2155];
%嵌套调用卷积函数 conv,计算 H(z) 分母的展开多项式系数向量 A
A=conv(conv(a1,a2),a3);
y=filter(B,A,x);            %对 x(n) 滤波
X=fft(x,N);                 %计算 x(n) 的 N 点 DFT
Y=fft(y,N);                 %计算 y(n) 的 N 点 DFT
[H,f]=freqz(B,A,N,'whole');       %计算滤波器的频率响应函数 H(e^jw)
Ym=H'.*X;                   %计算输出频谱 H(e^jw)·X(e^jw)
k=0:N-1;  f=2*k/N;
...
```

以下为绘图语句，用 stem(x, '.')和 stem(y, '.')语句分别画出输入输出的序列图 x(n)和 y(n)，用 plot(f, abs(X))和 plot(f, abs(Y))语句分别画出输入输出的幅频特性|X(jω)|和|Y(jω)|，另外，用 plot(f/pi, abs(H))和 plot(f/pi, abs(Ym)) 语句分别画出系统的幅频特性|H(jω)|及由频域算出的输出幅频特性|Ym(jω)|，标注语句从略。

■ **程序运行结果**

如图 7.14 所示。由图中时间序列和幅频曲线都可以看出，低通滤波器使输入信号 $x(n)$ 的高频成分得到很大衰减，让低频正弦信号和低频噪声通过输出。由于不是理想低通滤波器，所以在过渡带[0.2π, 0.4π]上，滤波器的幅频衰减随着频率升高而逐渐加大。比较|FT[x(n)]|和 FT[y(n)]|也可以看出这一点。

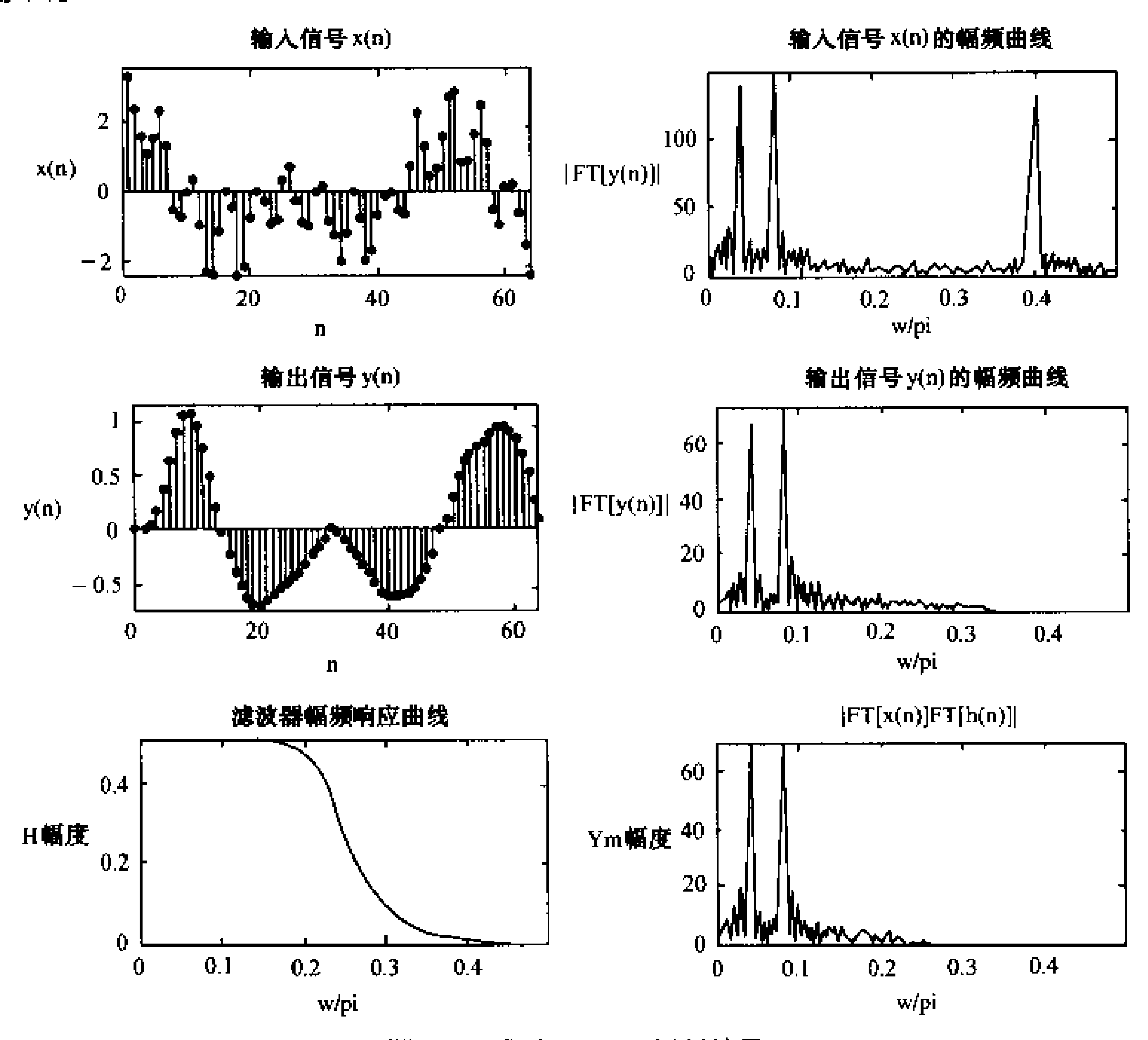


图 7.14　程序 q713.m 运行结果

图 7.14 中，$\left|Y(e^{j\omega})\right| = \left|FT[y(n)]\right| \approx \left|H(e^{j\omega})X(e^{j\omega})\right|$，即由时域卷积算出的输出频谱符合频域相乘的结果，证实了傅立叶变换的卷积时域定理。

【例 7.14】 用 symbolic(符号运算)工具箱解 z 变换问题

解：■ 分析

无限长度时间序列的 z 变换和逆 z 变换都属于符号运算的范围。MATLAB 的 symbolic（符号运算）工具箱已提供了这种函数。如果读者已在计算机上安装了这个工具箱，可以键入以下程序。

■ **MATLAB程序q714.m**

```
syms  z  n     a   N w0                                     % 规定 z,n,a 为符号变量
y1=a^n; Y1=ztrans(y1), pause                                % 给出 y1 的表示式,求其 z 变换 Y1
y2=n;     Y2=ztrans(y2), pause                              % 求其 z 变换 Y2
y3=n*a^n;  Y3=ztrans(y3)
y4=n*  (n-1)/2; Y4=ztrans(y4)
y5=exp(j*w0*n);  Y5=ztrans(y5)
y6=sin(w0*n),  Y6=ztrans(y6)
pause
X1= z/(z-1),x1 = iztrans(X1)           % 给出 X1 的表示式,求 X1 的逆 Z 变换 x1
X2 = -3*z^-1/(2-5*z^-1+2*z^-2),x2 = iztrans(X2)
X3 = z/(z-a),x3 = iztrans(X3)
X4 = z/((z-1)^2),x4 = iztrans(X4)
X5 = z/((z-1)^3),x5 = iztrans(X5)
X6 =  (1-z^-N)/(1-z^-1),x6 = iztrans(X6)
X7 = z/(z-exp(j*w0)),x7 = iztrans(X7)
```

■ **程序运行结果**

见表 7.3。

逆 z 变换的最后两行实际上不是答案，这说明符号运算工具箱还远未完善。

表 7.3　z 变换和逆 z 变换部分结果

	输入时间序列 y	z 变换 Y(z)=ztrans(y)
z 变换 ztrans	y1 = a^n	Y1 = -z/(-z+a)
	y2 = n	Y2 = z/(z-1)^2
	y3 = n*a^n	Y3 = z*a/(-z+a)^2
	y4 = 1/2*n*(n-1)	Y4 = z/(z-1)^3
	y5 = exp(I*w0*n)	Y5= z*(-cos(n)+z+i*sin(n))/(-2*z*cos(n)+z^2+1)
	y6 = sin(w0*n)	Y6 = sin(n)*z/(-2*z*cos(n)+z^2+1)
逆 z 变换 iztrans	输入 z 变换 X(z)	逆 z 变换 x(n) = iztrans(X)
	X1 = -3/z/(2-5/z+2/z^2)	x1 = -2^n+(1/2)^n
	X2 = z/(z-1)	x2 = 1
	X3 = z/(z-a)	x3 = a^n
	x4 = n	X4 = z/(z-1)^2
	X5 = z/(z-1)^3	x5 = -1/2*n+1/2*n^2
	X6 = (1-z^(-N))/(1-1/z)	X6 = iztrans((1-z^(-N))/(1-1/z), z, n)
	X7 = z/(z-exp(i*w0))	x7 = z*iztrans(1/(z-exp(i*w0)), w0, n)

7.3　离散傅立叶变换（DFT）

DFT 是数字信号处理中最重要的数学工具之一。其实质是对有限长序列频谱的离散化，即通过 DFT 使时域有限长序列与频域有限长序列相对应，从而可在频率域用计算机进行信号处理。更重要的是 DFT 有多种快速算法（FFT—Fast Fourier Transform），可使信号处理速度提高好几倍，使数字信号的实时处理得以实现。因此，DFT 既有重要的理论意义，又有广泛的实际应用价值。

熟悉 DFT 的定义、物理意义和重要性质，有助于正确使用 DFT 解决数字信号处理的实际问题。本节主要结合典型例题，用 MATLAB 工具箱函数，以序列的时域和频域波形直观地验证 DFT 的物理意义及频域采样理论。

先讨论离散傅立叶变换（DFT）的定义与 MATLAB 计算。

设序列 $x(n)$长度为 M，则 $x(n)$的 $N(N \geqslant M)$点离散傅立叶变换对定义为

$$X(k)=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{kn},\quad k=0,1,2,\cdots,N-1 \tag{7.3}$$

$$x(n)=\mathrm{IDFT}[X(k)]=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-kn},\quad n=0,1,2,\cdots,N-1 \tag{7.4}$$

其中，$W_N=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$，$N$ 称为 DFT 变换区间长度。

用类似于例 7.9 中的方法，可把（7.3）式写成矩阵乘法运算

$$X(k)=\boldsymbol{xn}\,\boldsymbol{Wnk} \tag{7.5}$$

其中，$\boldsymbol{xn}$ 为序列行向量，$\boldsymbol{Wnk}$ 是一 $N\times N$ 阶方阵，通常称之为旋转因子矩阵。

$$\boldsymbol{xn}=[x(0),x(1),x(2),\cdots,x(N-1)]$$

$$\boldsymbol{Wnk}=\begin{bmatrix} W_N^{0\times0} & W_N^{0\times1} & \cdots & W_N^{0\times(N-1)} \\ W_N^{1\times0} & W_N^{1\times1} & \cdots & W_N^{1\times(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^{(N-1)\times0} & W_N^{(N-1)\times1} & \cdots & W_N^{(N-1)\times(N-1)} \end{bmatrix} \tag{7.6}$$

（7.5）式可用 MATLAB 的矩阵运算表示为：

Wnk = WN.^ ([0:N−1]'*[0:N−1])

因此，可得到用矩阵乘法计算 N 点 DFT 的程序如下。

■ MATLAB程序q73a.m

```
%用矩阵乘法计算 N 点 DFT
clear; close all
xn=input('请输入序列 x= ');
N = length(xn);     %
n=0:N-1;  k=n;  nk=n'*k;   % 生成[0:N-1]'* [0:N-1]方阵
WN=exp(-j*2*pi/N);
Wnk=WN.^nk;                % 产生旋转因子矩阵
Xk=xn*Wnk;                 % 计算 N 点 DFT
```

只要输入序列 x(n)，运行该程序，即可实现 x(n)的 N 点 DFT。这种计算离散傅立叶变换的方法概念清楚，编程简单。但占用内存大，运行速度低，所以不实用。MATLAB 基础部分提供了 fft，ifft，fft2 和 ifft2 等快速计算傅立叶变换的函数，它使 DFT 的运算速度量提高了若干数量级。信号处理工具箱提供了 czt，dct，idct，fftshift 等有关的变换函数。为了简化程序，后面的例题均直接调用这些函数。

● fft 和 ifft 一维快速正逆傅立叶变换

➢ X = fft(x，N) 采用 FFT 算法计算序列向量 x 的 N 点 DFT。缺省 N 时 fft 函数自动按 x 的长度计算 DFT。当 N 为 2 的整数次幂时，fft 按基 2 算法计算，否则用混合基算法。ifft 的调用格式相仿。

● fft2 和 ifft2 二维快速正逆傅立叶变换

● czt 线性调频 z 变换

➢ y = czt(x, m, w, s) 它计算由 z = a*w.^(-(0:m-1))定义的 z 平面螺旋线上各点的 z 变换。可见 a 规定了起点，w 规定了相邻点的比例，m 规定了变换的长度。后三个变元缺省值是 a = 1，w = exp(j*2*pi/m)及 m = length(x)。因此，y = czt(x) 就等于 y = fft(x)。可键入 cztdemo 加深理解。

● dct 和 idct 正逆离散余弦变换

➢ y = dct(x，N) 可完成如下的变换：

$$y(k)=\sum_{n=1}^{N}2x(n)\cos\left(\frac{\pi}{2n}k(2n+1)\right) \qquad k=0,1,\cdots,N-1$$

N 的缺省值为 length(x)。

➢ Y = fftshift(X) 用来重新排列 X = fft(x)的输出，当 X 为向量时，它把 X 的左右两半进行交换。从而把零频分量移至频谱的中心。如果 X 是二维傅立叶变换的结果，它同时把 X 左右和上下进行交换。

➢ y = fftfilt(b, x) 采用重叠相加法 FFT 实现对信号向量 x 快速滤波，得到输出序列向量 y。向量 b 为 FIR 滤波器的单位脉冲响应序列，h(n) = b(n+1)，n = 0，1，2，…，length(b)-1。

➢ y = fftfilt(b, x, N) 自动选取 FFT 长度 NF = 2^nextpow2(N)，输入数据 x 分段长度 M = NF-length(b)+1。其中 nextpow2(N)函数求得一个整数，满足：

2^(nextpow2(N)-1)<N≤2^nextpow2(N)

缺省 N 时，fftfilt 自动选择合适的 FFT 长度 NF 和对 x 的分段长度 M。

【例 7.15】 基本序列的离散傅立叶变换计算

已知以下序列

复正弦序列 $x_1(n)=\mathrm{e}^{\mathrm{j}\frac{\pi}{8}n}\cdot R_N(n)$

余弦序列 $x_2(n)=\cos\left(\frac{\pi}{8}n\right)\cdot R_N(n)$

正弦序列 $x_3(n)=\sin\left(\frac{\pi}{8}n\right)\cdot R_N(n)$

分别对 N=16 和 N=8 计算以上序列的 N 点 DFT，并绘出幅频特性曲线，最后用 DFT 理论解释为何两种 N 值下的 DFT 结果差别如此之大。

解：直接产生序列 X1n，X2n 和 X3n，调用 fft 函数求解本题的程序 q715.m 如下。

■ **MATLAB程序q715.m**

```
% DFT 计算
clear;close all
N=16;  N1=8;
n=0:N-1;  k=0:N1-1;
%产生序列 x1(n),计算 DFT[x1(n)]
x1n=exp(j*pi*n/8);         %产生 x1(n)
X1k=fft(x1n,N);            %计算 N 点 DFT[x1(n)]
Xk1=fft(x1n,N1);           %计算 N1 点 DFT[x1(n)]
%产生序列 x2(n),计算 DFT[x2(n)]
x2n=cos(pi*n/8);
X2k=fft(x2n,N);            %计算 N 点 DFT[x2(n)]
Xk2=fft(x2n,N1);           %计算 N1 点 DFT[x1(n)]
%产生序列 x3(n),计算 DFT[x3(n)]
x3n=sin(pi*n/8);
X3k=fft(x3n,N);            %计算 N 点 DFT[x3(n)]
Xk3=fft(x3n,N1);           %计算 N1 点 DFT[x1(n)]
...
```

用 stem(n, abs(X1k), '.')，stem(n, abs(X2k), '.')，stem(n, abs(X3k), '.')，stem(k, abs(Xk1), '.')，stem(k, abs(Xk2), '.')，stem(k, abs(Xk3), '.')语句画出这三种信号、两种采样序列长度、共六种情况的曲线。图形划分及标注语句从略。

■ **程序运行结果**

如图 7.15 所示。

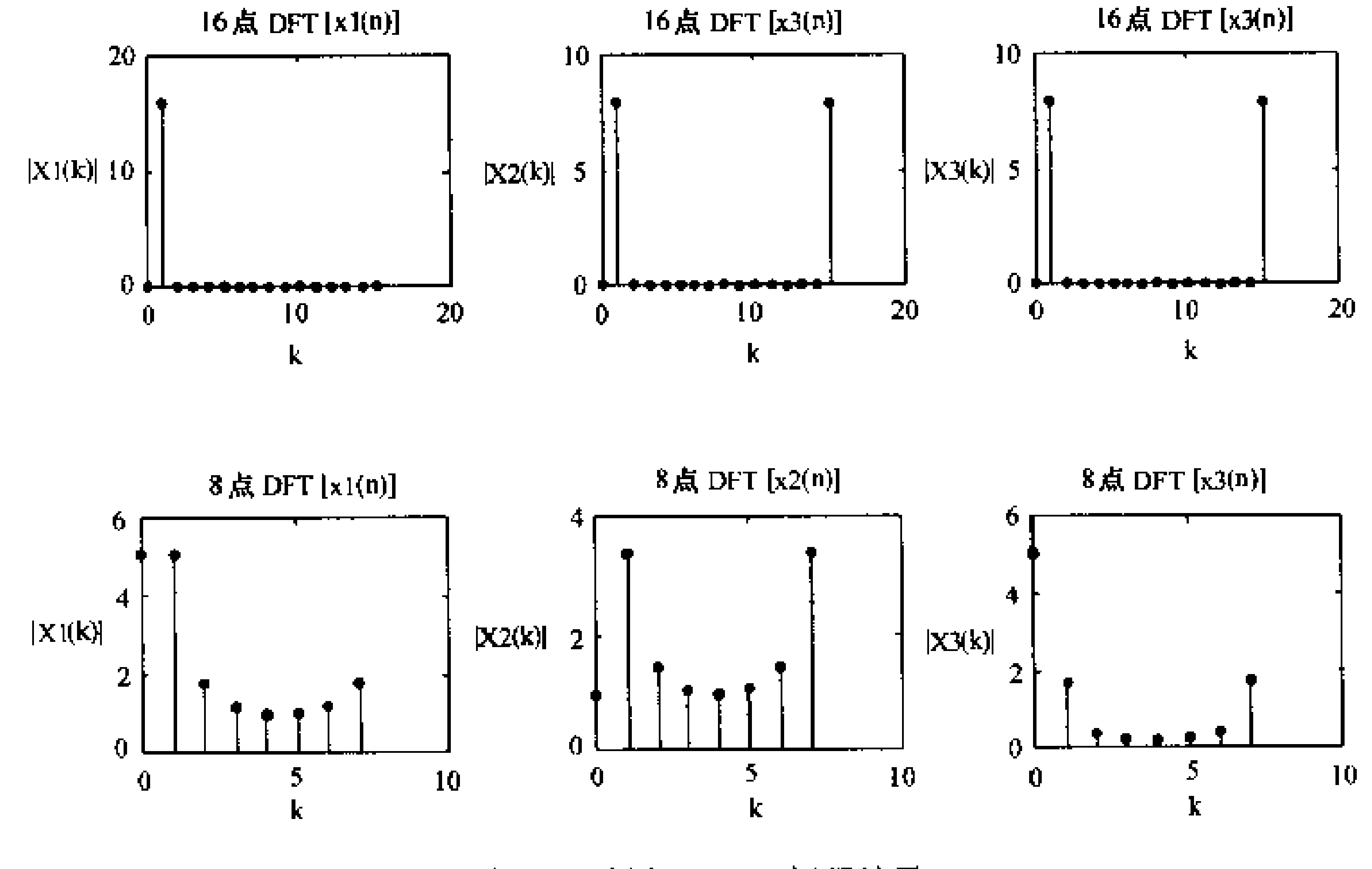


图 7.15　程序 q715.m 运行结果

N 点 DFT[x(n)] = X(k)的一种物理解释是，$X(k)$是 $x(n)$以 N 为周期的周期延拓序列的离散傅立叶级数系数 $\tilde{X}(k)$ 的主值区序列。即 $X(k)=\tilde{X}(k)R_N(k)$。

（1）N=16 时

$$x_1(n)=\mathrm{e}^{\mathrm{j}\frac{\pi}{8}n}R_N(n)\ ,\ x_2(n)=\cos\left(\frac{\pi}{8}n\right)R_N(n)\ ,\ x_3(n)=\sin\left(\frac{\pi}{8}n\right)R_N(n)$$

$x_1(n),x_2(n),x_3(n)$ 正好分别是正弦序列 $\mathrm{e}^{\mathrm{j}\frac{\pi}{8}n},\cos\left(\frac{\pi}{8}n\right),\sin\left(\frac{\pi}{8}n\right)$的一个周期。所以，$x_1(n),x_2(n),x_3(n)$ 的周期延拓序列就是这三个单一频率的正弦序列。其离散傅立叶级数的系数分别如图 7.15 中的 16 点 DFT[x1(n)]，DFT[x2(n)]和 DFT[x3(n)]所示。

（2）N=8 时

$x_1(n),x_2(n),x_3(n)$ 正好分别是正弦序列 $\mathrm{e}^{\mathrm{j}\frac{\pi}{8}n},\cos\left(\frac{\pi}{8}n\right),\sin\left(\frac{\pi}{8}n\right)$的半个周期。所以，$x_1(n),x_2(n),x_3(n)$ 以 N 为周期的周期延拓序列不再是单一频率的正弦序列（例如，$x_2((n))_N=\left|\cos\left(\frac{\pi}{8}n\right)\right|$），其中含有丰富的谐波成分，其离散傅立叶级散系数与 $N=16$ 时差别很大。为此，对周期信号进行谱分析时，一定要截取整数个周期。否则，得到的将是错误的频谱。为了说明这点，在运行 q715.m 后，可随即键入

```
subplot(2, 1, 1), stem(n, x2n(n+1), '.');
subplot(2, 1, 2), stem(k, x2n(k+1), '.')
```

不难看出，在截取 16 点时，得到的是完整的余弦波形；而截取 8 点时，得到的是半截的余弦波形，当然它有大量的谐波成分。

【例 7.16】 验证 N 点 DFT 的物理意义

$$X(k)=\mathrm{DFT}\ [x(n)]=X(\mathrm{e}^{\mathrm{j}\omega})|_{\omega=\frac{2\pi}{N}k}\ ,\quad (k=0,\ 1,\ \cdots,\ N-1)$$

（1）$x(n)=R_4(n)$，$X(\mathrm{e}^{\mathrm{j}\omega})=\mathrm{FT}[x(n)]=\dfrac{1-\mathrm{e}^{-\mathrm{j}4\omega}}{1-\mathrm{e}^{-\mathrm{j}\omega}}$，绘出幅频曲线和相频曲线。

（2）计算并图示 $x(n)$的 8 点 DFT。

（3）计算并图示 $x(n)$的 16 点 DFT。

解：■ 建模

长度为 M 的序列 $x(n)$的傅立叶变换和 N（$N\geqslant M$）点离散傅立叶变换的定义式如下

$$X(\mathrm{e}^{\mathrm{j}\omega})=\mathrm{FT}[X(n)]=\sum_{n=0}^{N-1}x(n)\mathrm{e}^{-\mathrm{j}\omega n} \tag{7.7}$$

$$X(k)=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{kn}\quad (k=0,1,2,\cdots,N-1) \tag{7.8}$$

式中，$W_N=\mathrm{e}^{-\frac{\mathrm{j}2\pi}{N}}$。比较（7.8）式和（7.7）式可得出 $x(n)$的傅立叶变换与 N（$N\geqslant M$）点离散傅立叶变换的关系式为

$$X(k)=\mathrm{DFT}[x(n)]=X(\mathrm{e}^{\mathrm{j}\omega})|_{\omega=\frac{2\pi}{N}k}\quad (k=0,1,2,\ \cdots,N-1) \tag{7.9}$$

简言之，序列 $x(n)$的 N点 DFT 的物理意义是对 $X(\mathrm{e}^{\mathrm{j}\omega})$ 在[0，2π]上进行 N点等间隔采样。

为了验证这种关系，程序 q716.m 先对$|X(e^{j\omega})|$密集采样，绘制出幅频曲线图。然后再分别做 8 点和 16 点 DFT 来验证如上采样关系。为了能直观地看出 $X(k)$与$X(e^{j\omega})$之间的采样关系，在 $X(k)$的幅度和相位图中同时画出了$X(e^{j\omega})$的幅频和相频曲线。

■ **MATLAB程序q716.m**

```
% 计算并图示 DFT[x(n)]
clear;close all
N1=8;N2=16;                    % 两种 FFT 变换长度
n=0:N1-1;
w=2*pi* (0:2047)/2048;
Xw=(1-exp(-j*4*w))./(1-exp(-j*w));     % 对 x(n)的频谱函数采样 2048 点
subplot(3, 2, 1);plot(w/pi, abs(Xw))
title('x(n)的幅频曲线');xlabel('ω/π');ylabel('幅度');
subplot(3, 2, 2);plot(w/pi, angle(Xw))
title('x(n)的相频曲线');axis([0, 2, -pi, pi]);line([0, 2], [0, 0])
xlabel('ω/π');ylabel('相位 (rad)');
xn=[(n>=0)&(n<4)]; % 产生 x(n)=xn
X1k=fft(xn, N1);  %计算 N1 点 DFT[x(n)]
X2k=fft(xn, N2);  %计算 N2 点 DFT[x(n)]
figure(2)
k1=0:N1-1;
subplot(3, 2, 1);stem(k1, abs(X1k), '.')     % 画 N1 点离散频谱幅度
title('N1 点 DFT[x(n)]=X1(k)')
xlabel('k  (ω=2πk/N1)');ylabel('|X1(k)|');
hold on
plot(N1/2*w/pi, abs(Xw))                    % 叠加上连续频谱幅度
subplot(3, 2, 2);stem(k1, angle(X1k), '.')    % 画 N1 点离散频谱相位
title('X1(k)的相位');axis([0, N1, -pi, pi]);line([0, N1], [0, 0])
xlabel('k(ω=2πk/N1)');ylabel('相位 (rad)');
hold on
plot(N1/2*w/pi, angle(Xw))                  % 叠加上连续频谱相位
……
```

可同样绘制变换长度为 N2 时的 X2(k)的振幅和相位曲线，程序从略。

■ **程序运行结果**

如图 7.16 所示。可看出 N 点 DFT [x(n)] 确实是对$X(e^{j\omega})$在 [0，2π] 区间内的 N 点等间隔采样。

【例 7.17】 验证频域采样与时域采样的对偶性

试编写 MATLAB 程序，来验证对频谱$X(e^{j\omega})$进行等间隔采样。对应于时域序列周期延拓，要求采取如下做法。

（1）产生一个三角波序列 $x(n)$

$$x(n)=\begin{cases} n & 0\leqslant n\leqslant M/2 \\ M-n & M/2\leqslant n\leqslant M \end{cases}$$

（2）对 $M=40$，计算 $x(n)$的 64 点 DFT，并图示 $x(n)$和 $X(k)=\text{DFT}[x(n)]$，$k=0, 1, \cdots, 63$。

（3）对（2）中所得 $X(k)$在 $[0, 2\pi]$ 上进行 32 点抽样得

$$X_1(k)=X(2k)\text{，}k=0,\ 1,\ \cdots,\ 31$$

（4）求 $X_1(k)$ 的 32 点 IDFT，即　$x_1(n)=\text{IDFT}[X_1(k)]$，$k=0, 1, \cdots, 31$

（5）绘出 $x_1((n))_{32}$ 的波形图，评述 $x_1((n))_{32}$ 与 $x(n)$ 的关系。并根据频域采样理论加以解释。

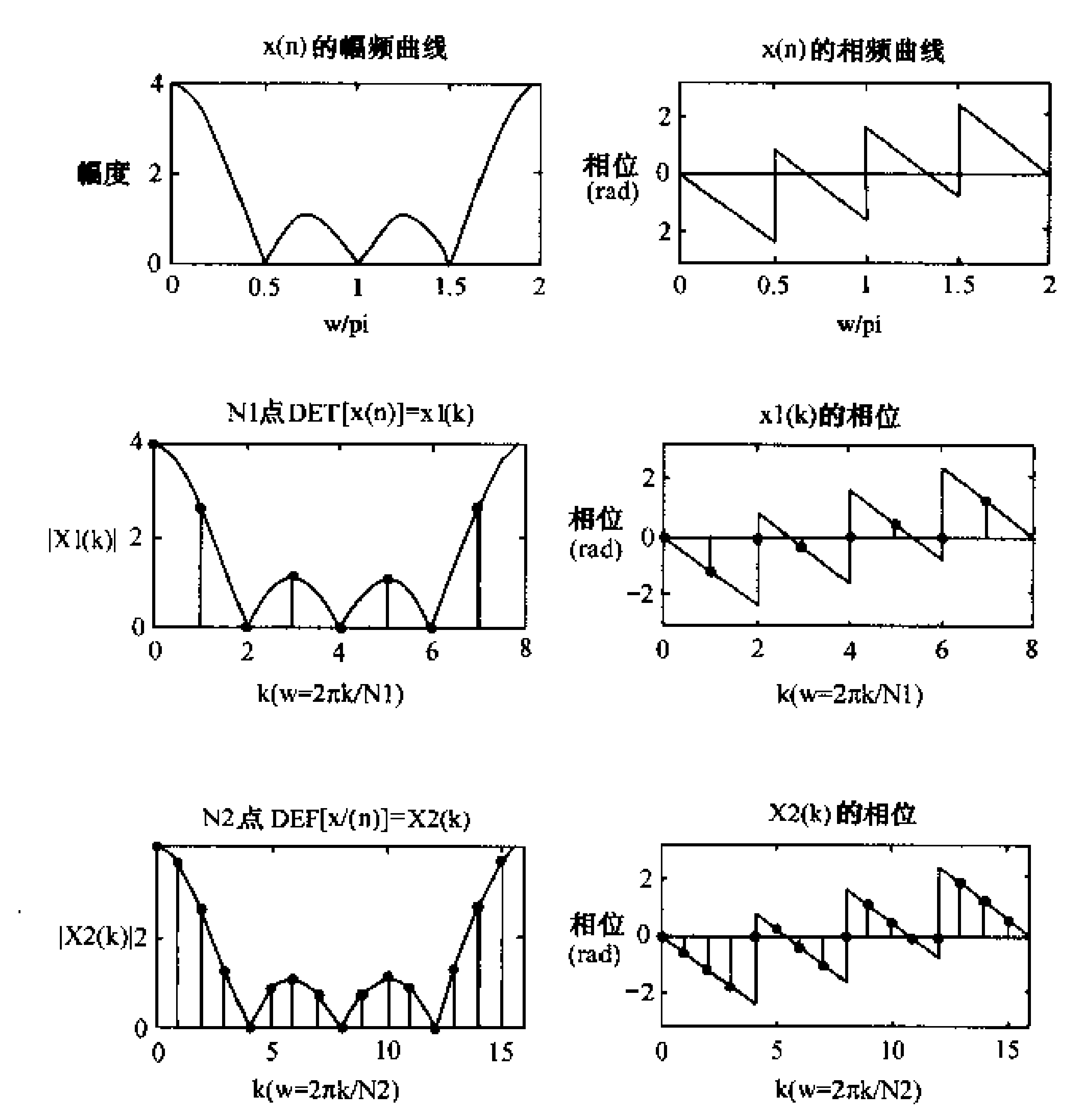


图 7.16　离散傅立叶变换与傅立叶变换的采样关系

解：■ 建模

我们知道，序列 $x(n)$的傅立叶变换 $X(e^{j\omega})=\text{FT}[x(n)]$是以 2π为周期的。所以对其以 $2\pi/N$ 为间隔采样，得到以 N 为周期的频域序列 $\tilde{X}(k)=X(e^{j(\frac{2\pi}{N}k)})$。根据离散傅立叶级数理论，$\tilde{X}(k)$ 应该是一个以 N 为周期的序列 $\tilde{x}(n)$ 的离散傅立叶级数系数。即

$$\tilde{x}(n)=\sum_{k=0}^{N-1}\tilde{X}(k)W_N^{-kn}=\sum_{m=-\infty}^{\infty}x(n+mN) \tag{7.10}$$

$\tilde{x}(n)$ 是原序列 $x(n)$以 N 为周期的周期延拓序列。只要求出 $\tilde{x}(n)$的主值区序列 $x_1(n)$，将其周期延拓，就得到 $\tilde{x}(n)$。对本题而言

$$X_1(k)=\widetilde{X}(k)R_N(k)\text{，}N=32$$

$$x_1(n)=\mathrm{IDFT}[X_1(k)]$$

其中，$X_1(k)$可用题中所给的方法产生。

■ **MATLAB程序q717.m**

```
% 时域与频域采样的对偶性验证
clear;close all
M=40;  N=64;   n=0:M;
xa=0:floor(M/2);    xb= ceil(M/2)-1:-1:0;
xn=[xa,xb];               % 产生长度为 M 的三角波序列 xn
Xk=fft(xn,64);            % 计算 xn 的 64 点 FFT[x(n)]
X1k=Xk(1:2:N);            % 隔点抽取 Xk 得到 X1(K)
x1n=ifft(X1k,N/2);        % 计算 X1k 的 32 点 IFFT[X1(k)]得到 x1(n)
nc=0:3*N/2;               % 取 97 点进行观察
xc=x1n(mod(nc,N/2)+1);    %x1(n)的周期延拓序列
……
   stem(n,xn,'.'),
   k=0:N-1;  stem(k,abs(Xk),'.')
   k=0:N/2-1;  stem(k,abs(X1k),'.')
   n1=0:N/2-1;  stem(n1,x1n,'.')
   stem(nc,xc,'.')
```

用以上五行语句画出这五种情况的曲线。图形划分及标注语句从略。

■ **程序运行结果**

如图 7.17 所示。图中 $x1((n))_{32}$ 满足（7.9）式。从而验证了频域对 $X(e^{j\omega})$ 以间隔 $2\pi/32$ 采样，时域对应原序列以 32 为周期的周期延拓。

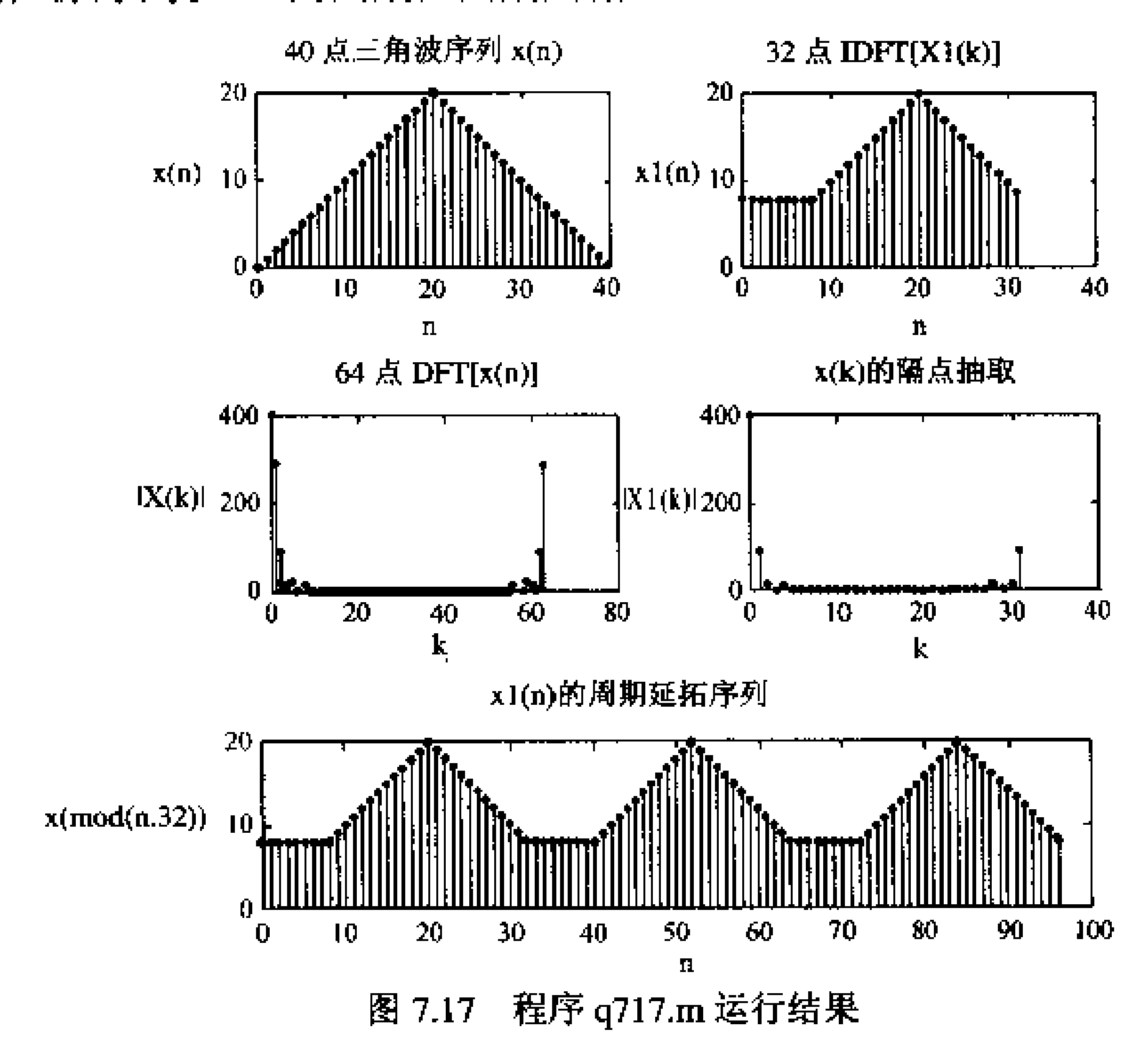


图 7.17　程序 q717.m 运行结果

由于频域在[0，2π]上的采样点数 N（$N = 32$）小于 $x(n)$的长度 M（$M = 40$），所以，产生时域混叠现象，不能由 $X_1(k)$恢复原序列 $x(n)$。只有满足 $N \geqslant M$ 时，可由频域采样 $X_1(k)$得到原序列 $x(n)$。

$$x(n) = \mathrm{IDFT}[X_1(k)]$$

这就是频域采样定理。对 $N \geqslant M$ 的情况，请读者自己编程验证。

【例 7.18】 快速卷积

用快速卷积法计算下面两个序列的卷积。并测试直接卷积和快速卷积的时间。

$$x(n) = 0.9^n R_M(n)$$
$$h(n) = R_N(n)$$

解：■ 建模

数字滤波器对输入信号 $x(n)$进行滤波处理就是计算其单位脉冲响应 $h(n)$与输入信号 $x(n)$的线性卷积。所以，计算卷积的速度直接影响滤波器的处理速度。快速卷积就是根据 DFT 的循环卷积性质，将时域卷积转换为频域相乘，最后再进行 IDFT 得到时域卷积序列 $y(n)$。其中时域和频域之间的变换均用 FFT 实现，所以使卷积速度大大提高。快速卷积计算框图如图 7.18-1 所示。按照该框图很容易编写出程序 q718.m。

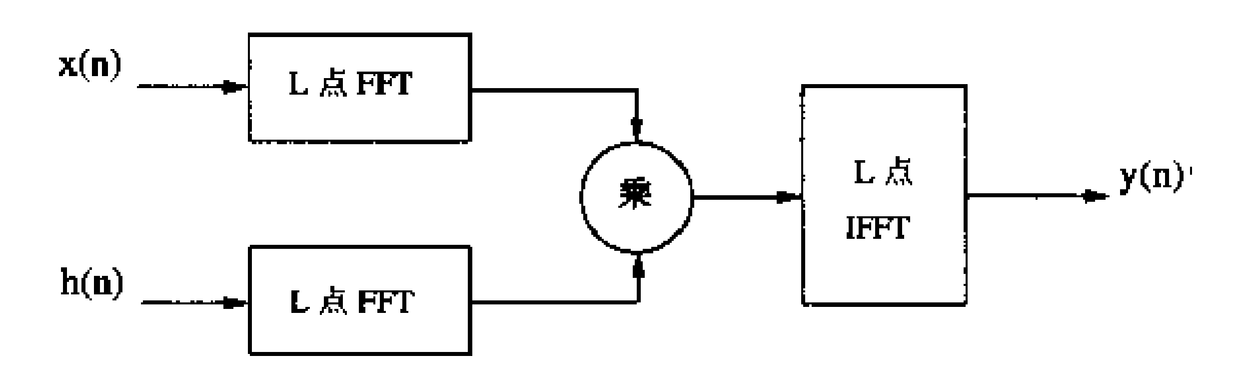


图 7.18-1　快速卷积框图

■ MATLAB程序q718.m

```
clear;close all
xn=input('请输入 x(n)序列: xn= 书上用 sin(0.4* [1:15])');
hn=input('请输入 h(n)长度: hn= 书上用 0.9.^[1:20]');
M=length(xn); N=length(hn);
nx=1:M; nh=1:N;
%循环卷积等于线性卷积的条件: 循环卷积区间长度 L>=M+N-1
L=pow2(nextpow2(M+N-1));          % 取 L 为大于等于且最接近 (N+M-1) 的 2 的正次幂
tic,                              % 快速卷积计时开始
Xk=fft(xn,L);                     % L 点 FFT[x(n)]
Hk=fft(hn,L);                     % L 点 FFT[h(n)]
Yk=Xk.*Hk;                        % 频域相乘得 Y(k)
yn=ifft(Yk,L);                    % L 点 IFFT 得到卷积结果 y(n)
toc                               % 快速卷积计时结束
subplot(2,2,1),stem(nx,xn,'.');
ylabel('x(n)')
subplot(2,2,2),stem(nh,hn,'.');
```

```
ylabel('h(n)')
subplot(2,1,2);ny=1:L;
stem(ny,real(yn),'.');ylabel('y(n)')
tic,yn=conv(xn,hn); toc        %直接调用函数 conv 计算卷积与快速卷积比较
```

■ **程序运行结果**

按提示输入 sin(0.4*[1:15])及 h= 0.9.^(1:20)后，输出结果如图 7.18-2 所示。图 7.18-1 中，FFT 变换的长度 L 必须满足 L≥N+M-1，输出 y(n)才等于 x(n)和 h(n)的线性卷积。

因为 MATLAB 中的计时比较粗糙，要比较两种算法的运行时间，必须取较大的 N 和 M。在作者的计算机上，当 N = M = 4096 时，快速卷积的执行时间为 0.11s，直接调用卷积函数 conv 计算卷积的时间为 2.09s。应当注意，以上结果是把 FFT 变换区间长度 L 取为大于或等于（N+M-1）的 2 的最小整次幂时得出的。若不这样取，快速卷积的执行时间可能会增大十几倍。读者可以试试将程序中的 FFT 计算改成直接计算 DFT 的矩阵乘运算，测试一下需要执行多少时间。

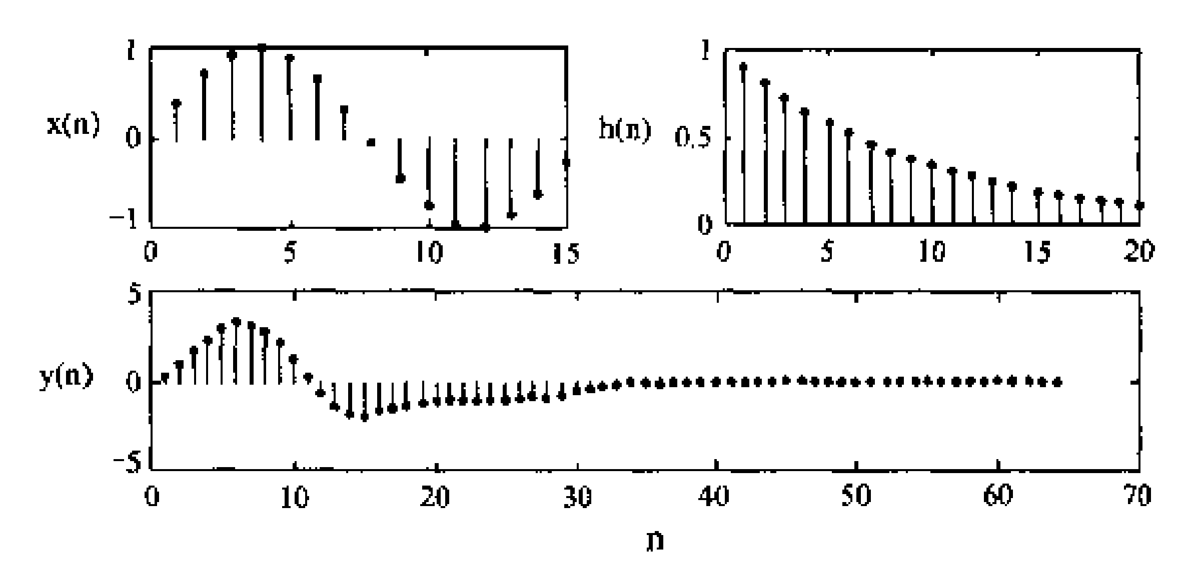


图 7.18-2　$x(n)$，$h(n)$及其线性卷积波形（N=M=30）

【例 7.19】 用 DFT 对连续信号作谱分析

已知 $x_a(t)=\cos(200\pi t)+\sin(100\pi t)+\cos(50\pi t)$

用 DFT 分析 $x_a(t)$ 的频谱结构。选择不同的截取长度 T_p，观察用 DFT 进行频谱分析时存在的截断效应（频谱泄露和谱间干扰）。试用加窗的方法减少谱间干扰。

解：■ 建模

在计算机上用 DFT 对模拟信号进行谱分析时，只能以有限大的采样频率 f_s 对模拟信号采样有限点样本序列（等价于截取模拟信号一段进行采样）作 DFT 变换，得到模拟信号的近似频谱。其误差主要来自以下因素：

① 截断效应（频谱泄露和谱间干扰）

② 频谱混叠失真

因素①使谱分辨率（能分辨开的两根谱线间的最小间距）降低，并产生谱间干扰；因素②使折叠频率（$f_s/2$）附近的频谱产生较大失真。理论和实践都已证明，加大截取长度 T_p 可提高频率分辨率；选择合适的窗函数可降低谱间干扰；而频谱混叠失真要通过提高采样频率 f_s 和（或）预滤波（在采样之前滤除折叠频率以外的频率成分）来改善。

按题目要求编写程序 q719.m 验证截断效应及加窗的改善作用，先选取以下参数：

- 采样频率 $f_s=400$Hz，$T=1/f_s$
- 采样信号序列 $x(n)=x_a(nT)w(n)$，　n=0，1，2，…，N-1

● 对$x(n)$作4096点DFT作为$x_a(t)$的近似频谱$X_a(jf)$。

其中，N为采样点数，$N=f_sT_p$，T_p为截取时间长度，$w(n)$称为窗函数。

取三种长度①T_p=0.04s，② T_p=4*0.04，③ T_p=8*0.04 和两种窗函数①矩形窗函数$w(n)=R_N(n)$，②Hamming 窗 $w(n)=\left[0.54-0.46\cos\left(\frac{2\pi n}{N-1}\right)\right]R_N(n)$。由程序中调用函数 hamming 产生。wn = hamming(N)产生长度为N的hamming窗函数列向量wn。

为了缩短程序，对三种 T_p 采用了循环语句，读者可注意循环中调用不同标注的技巧(st赋值语句)。其实对两种窗函数，也可再加一重循环语句，进一步缩短程序。

■ **MATLAB程序q719.m**

```
clear;close all
fs=400; T=1/fs;     %采样频率为400Hz
Tp=0.04; N=Tp*fs;   %采样点数N
N1=[N, 4*N, 8*N];                              % 设定三种截取长度供调用
st=['|X1(jf)|';'|X4(jf)|';'|X8(jf)|'];         % 设定三种标注语句供调用
%矩形窗截断
for m=1:3
   n=1:N1(m);
   %产生采样序列x(n)
     xn=cos(200*pi*n*T)+sin(100*pi*n*T)+cos(50*pi*n*T);
   Xk=fft(xn,4096);  %4096点DFT,用FFT实现
   fk=[0:4095]/4096/T;
   subplot(3,2,2*m-1)
   plot(fk,abs(Xk)/max(abs(Xk)));ylabel(st(m,:))
   if m==1 title('矩形窗截取');end
end
%加hamming窗改善谱间干扰
for m=1:3
   n=1:N1(m);
   wn=hamming(N1(m)); %调用工具箱函数hamming产生N长hamming窗序列wn
   xn=(cos(200*pi*n*T)+sin(100*pi*n*T)+cos(50*pi*n*T)).*wn';
   Xk=fft(xn,4096);  %4096点DFT,用FFT实现
   fk=[0:4095]/4096/T;
   subplot(3,2,2*m)
   plot(fk,abs(Xk)/max(abs(Xk)));ylabel(st(m,:))
   if m==1 title('Hamming窗截取');end
end
```

■ **程序运行结果**

如图7.19所示。图中X1(jf)，X4(jf)和X8(jf)分别表示Tp = 0.04s，0.04*4s和0.04*8s对的谱分析结果（运行程序对只输入0.04，其他两种Tp值由程序内部自动计算产生）。由图可见，由于截断使原频谱中的单频谱线展宽（又称之为泄漏），Tp越大泄漏越小，频率分辨

率越高。Tp = 0.04s 时，25Hz 与 50Hz 两根谱线已分辨不清了。所以实际谱分析的截取时间 Tp 是由频率分辨率决定的。另外，在本应为零的频段上出现了一些参差不齐的小谱包（称为谱间干扰）。谱间干扰的大小取决于加窗的类型。

比较加矩形窗和 Hamming 窗的谱分析结果可见，用矩形窗比用 Hamming 窗的频率分辨率高（泄漏小），但谱间干扰刚好相反。所以 Hamming 窗以牺牲分辨率换来谱间干扰的降低。实际上这是一般规律。更详细的分析以及各种窗函数性能见数字信号处理的书。读者可选用其他窗函数作比较。并用傅立叶变换理论给出产生截断效应的数学解释。

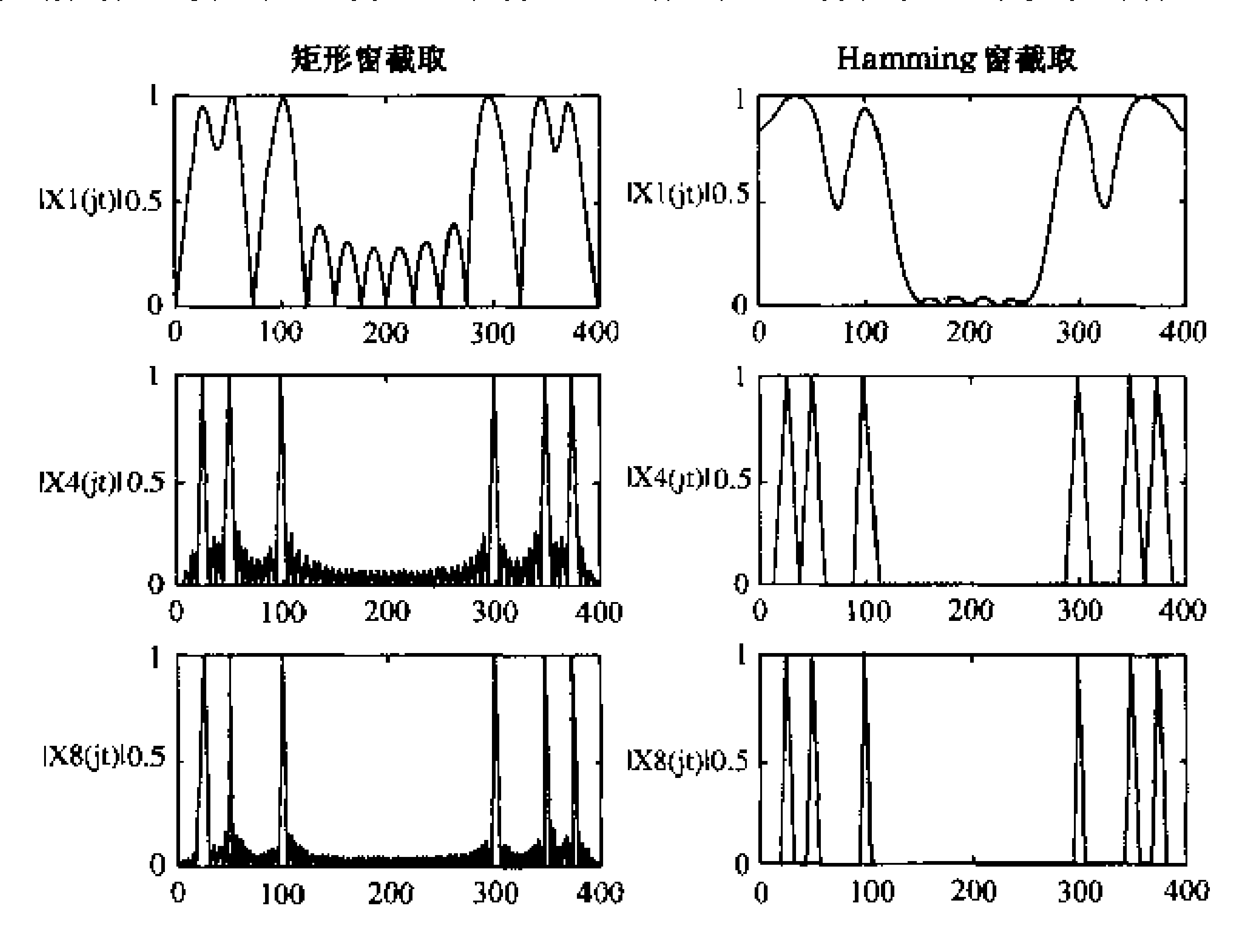


图 7.19　程序 q719.m 运行结果

7.4　数字滤波器结构

要将数字滤波器 $H(z)$用于处理数字信号，就必须构造出合适的实际实现结构（硬件实现结构或软件运算结构）。而且，对同样的系统函数 $H(z)$有几种不同的实现结构，各种结构的性能也不相同。常用的有以下几种。

（1）直接型(dir)结构：它对应于传递函数(tf)形式

$$H(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\cdots+b_Mz^{-M}}{a_0+a_1z^{-1}+a_2z^{-2}+\cdots+a_Nz^{-N}} \tag{7.11}$$

当 $a_0=1$ 时，称为标准形式。

（2）级联型(cas)结构（二阶分割(sos)形式）：它对应于零极增益形式，但为了避免复系数出现，采用了二阶分割，即将共轭零点和共轭极点分别组成实系数二阶环节。

$$H(z)=g\prod_{k=1}^{L}\frac{b_{c0k}+b_{c1k}z^{-1}+b_{c2k}z^{-2}}{1+a_{c1k}z^{-1}+a_{c2k}z^{-2}} \tag{7.12}$$

（3）并联型(par)结构：它对应于部分分式(residue)形式，为了避免复系数出现，也采用了二阶分割，即将共轭极点组成分母上的实系数二阶环节，留数则变为一阶环节。

$$H(z)=\sum_{k=1}^{L}\frac{b_{pk0}+b_{pk1}z^{-1}}{1+a_{pk1}z^{-1}+a_{pk2}z^{-2}}+\underbrace{\sum_{k=0}^{M-N}c_{pk}z^{-k}}_{\text{仅当}M\geq N\text{时}} \tag{7.13}$$

当 N = 偶数时，$L = N/2$。N = 奇数时，$L = (N-1)/2$，（7.12）和（7.13）式中有一个一阶分式。

（4）格型和格型梯型：其数学表述将在后面介绍。

实际中常常需要转换滤波器的结构形式，这种转换用手工计算非常复杂。MATLAB 提供了一系列的转换命令，使得复杂的转换问题用一条命令就可以实现。把这些命令列成表 7.4。由于这些结构形式与线性系统表述类型还有一定联系，所以把表述类型也都列入了表中。

表 7.4　线性系统类型和滤波器结构之间转换的工具箱函数表

	传递函数 num，den	状态空间 A，B，C，D	零极增益 z，p，k	部分分式 r，p，h	格型结构 K，V	级联结构 sos	并联结构
传递函数 num，den		tf2ss	tf2zp roots	residue residuez	tf2latc	tf2sos	dir2par*
状态空间	ss2tf		ss2zp			ss2sos	
零极增益	zp2tf poly	zp2ss				zp2sos	
部分分式 r，p，h	residue residuez						
格型结构 K，V	latc2tf						
级联结构 sos	sos2tf	sos2ss	sos2zp				
并联结构	par2dir*						

最常用的是从直接型转换到级联型和并联型，FIR 滤波器还需要将直接型转换成格型结构。表中左边的一列表示原始形式，上面的一行表示目标形式，交点处的函数名称就表示转换命令。这里是以 1998 年的 R11 版本为准，更低的版本中信号处理工具箱要少几种。带*号的函数不属于 MATLAB 的信号处理工具箱，它们引自文献[7]，本书提供了它的源程序。为了尊重原作者的工作，这里保留了它们的函数名，所以显得和其他转换函数不太统一。

MATLAB 还提供了各种类型滤波器实现滤波时的计算函数。比如

➢ filter 函数　用来计算直接型滤波器的输出 y = filter(B, A, x)，这在 7.1 节中已介绍过。

➢ filtic 函数　用来把过去的 x，y 数据转化为初始条件。

➢ sosfilt 函数　用来计算二阶分割型滤波器的输出 y = sosfilt(sos, x)。

➢ latcfilt 函数　用来计算格型滤波器的输出[F，G] = latcfilt (K, x)，其中 F 为前向格型滤波的输出，G 为后向格型滤波的输出。

下面结合例题介绍实现这几种转换的编程方法。

【例 7.20】 IIR 滤波器直接型到级联型和并联型转换

滤波器的系统函数为

$$H(z)=\frac{1-3z^{-1}+11z^{-2}-27z^{-3}+18z^{-4}}{16+12z^{-1}+2z^{-2}-4z^{-3}-z^{-4}}$$

求其级联型和并联型结构。

解：■ 建模

滤波器各种结构的相互转换实质上就是滤波器系统函数 $H(z)$的各种表示形式的相互转换。因此先写出 IIR 滤波器的直接型结构、级联型结构和并联型结构的 $H(z)$表示形式，则结构转换的 MATLAB 编程模型就清楚了。

由此可见，由直接型结构到级联型结构的转换，就是由（7.11）式所示的 $H(z)$的分子和分母多项式系数向量 $\boldsymbol{B}$ 和 $\boldsymbol{A}$ 求（7.12）式中的 L 个二阶子系统函数的分子和分母多项式系数矩阵 ***Bc*** 和 ***Ac***（***Bc*** 和 ***Ac*** 均为 $L\times3$ 阶矩阵）。而由直接型结构到并联型结构的转换，就是由（7.11）式所示的 $H(z)$的分子和分母多项式系数向量 $\boldsymbol{B}$ 和 $\boldsymbol{A}$ 求（7.13）式中的 L 个二阶子系统函数的分子和分母多项式系数矩阵 ***Bp*** 和 ***Ap*** 以及 FIR 部分的多项式系数向量 $\boldsymbol{C}$。***Bp*** 为 $L\times2$ 阶矩阵，***Ap*** 为 $L\times3$ 阶矩阵，***Cp*** 为（$M-N+1$）维行向量。

直接型和并联型中的二阶子系统均用直接型结构实现。

程序 q720.m 调用了信号处理工具箱函数 tf2sos 和扩展函数 dir2par，dir2par 中又调用了复共轭对比较函数 cplxcomp。由于 dir2par 和 cplxcomp 是文献[7]中开发的，不属于 MATLAB 工具箱函数，所以将其 M 文件清单附在程序 q720.m 之后，读者可将它存入自己的子目录中，以备调用。

➢ [sos, g] = tf2sos(B, A)　　实现从直接型到级联型（二阶分割形式）的转换。g 为（7.12）式中的增益，sos 为 L×6 阶矩阵，表示（7.12）式中的系数。

$$\text{sos}=\begin{bmatrix} b_{c01} & b_{c11} & b_{c21} & 1 & a_{c11} & a_{c21} \\ b_{c02} & b_{c12} & b_{c22} & 1 & a_{c12} & a_{c22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{c0L} & b_{c1L} & b_{c2L} & 1 & a_{c1L} & a_{c2L} \end{bmatrix}$$

➢ [Cp, Bp, Ap] = dir2par(B, A)　　实现从直接型到并联型的转换。B 为直接型 H(z)的分子多项式系数向量，A 为直接型 H(z)的分母多项式系数向量；Cp，Bp，Ap 的含义与扩展函数 dir2par 中的 C，B，A 相同。

■ MATLAB程序q720.m

```
%直接型到级联型和并联型转换
b=[1,-3,11,-27,18];
a=[16,12,2,-4,-1];
fprintf('级联型结构系数:')
[sos,g]=tf2sos(b,a)              %求级联型结构系数
fprintf('并联型结构系数:')
[Cp,Bp,Ap]=dir2par(b,a)          %求并联型结构系数
```

■ 程序运行结果

级联型结构系数

```
sos =  1.0000   -3.0000    2.0000    1.0000   -0.2500   -0.1250
       1.0000   -0.0000    9.0000    1.0000    1.0000    0.5000
g =    0625
```

并联型结构系数

```
Cp =   -18
Bp =   -10.0500    -3.9500
```

```
          28.1125    -13.3625
Ap =      1.0000      1.0000     0.5000
          1.0000     -0.2500    -0.1250
```

由级联型结构系数写出 $H(z)$表达式如下：

$$H(z)=0.0625\left(\frac{1+9z^{-2}}{1+z^{-1}+0.5z^{-2}}\right)\left(\frac{1-3z^{-1}+2z^{-2}}{1-0.25z^{-1}-0.125z^{-2}}\right)$$

级联型结构图如图 7.20-1 所示。

由并联型结构系数写出 $H(z)$表达式如下

$$H(z)=-18+\frac{-10.05-3.95z^{-1}}{1+z^{-1}+0.5z^{-2}}+\frac{28.1125-13.3625z^{-1}}{1-0.25z^{-1}-0.125z^{-2}}$$

并联型结构图如图 7.20-2 所示。

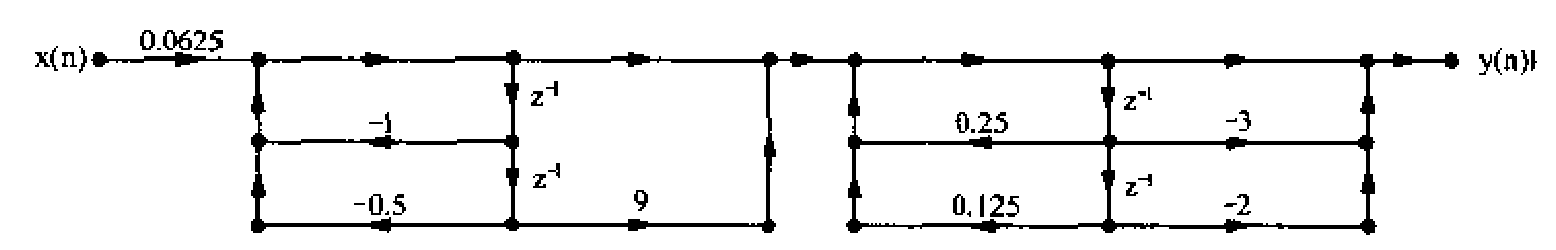


图 7.20-1　级联型结构图

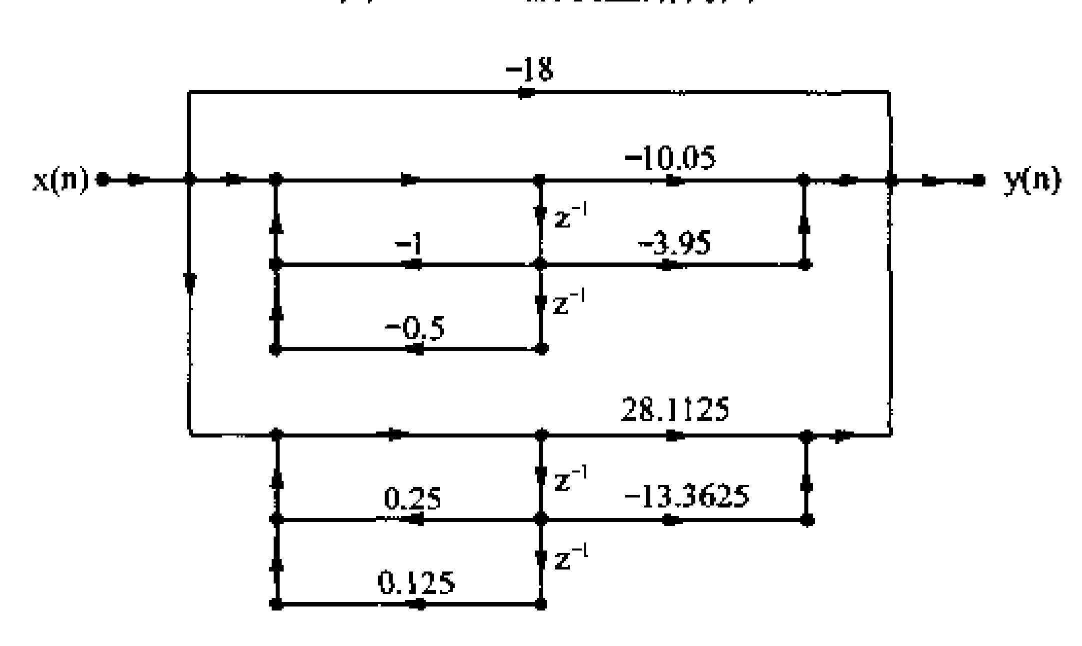


图 7.20-2　并联型结构

扩展函数 dir2par 的 M 文件(dir2par.m)清单

```
function [C, B, A] = dir2par(b, a)
% 直接型到并联型的转换
% ---------------------------------------
% [C,B,A] = dir2par(b,a)
%  C = 当 B 比 A 长时的多项式部分
%  B = 包含各 bk 的 K 乘 2 阶实系数矩阵
%  A = 包含各 ak 的 K 乘 3 阶实系数矩阵
%  b = 直接型的分子多项式系数
%  a = 直接型的分母多项式系数
%
M = length(b); N = length(a);
[r1,p1,C] = residuez(b,a);
```

```
p = cplxpair(p1,10000000*eps);
I = cplxcomp(p1,p);
r = r1(I);
K = floor(N/2); B = zeros(K,2); A = zeros(K,3);
if K*2 == N;  % N为偶时, A(z)的阶数为奇,有一个一阶环节
   for i=1:2:N-2
       Brow = r(i:1:i+1,:);
       Arow = p(i:1:i+1,:);
       [Brow,Arow] = residuez(Brow,Arow,[]);
       B(fix((i+1)/2),:) = real(Brow);
       A(fix((i+1)/2),:) = real(Arow);
       end
       [Brow,Arow] = residuez(r(N-1),p(N-1),[]);
       B(K,:) = [real(Brow) 0]; A(K,:) = [real(Arow) 0];
else
for i=1:2:N-1
        Brow = r(i:1:i+1,:);
        Arow = p(i:1:i+1,:);
        [Brow,Arow] = residuez(Brow,Arow,[]);
        B(fix((i+1)/2),:) = real(Brow);
        A(fix((i+1)/2),:) = real(Arow);
    end
end
```

复共轭对比较函数 cplxcomp.m 程序清单。

```
function I = cplxcomp(p1,p2)
%  I = cplxcomp(p1,p2)
% 比较两个模值相同但（可能）下标不同的复数对
% 本程序必须用在 CPLXPAIR 程序之后
% 排序极点向量及其相应的留数向量
%     p2 = cplxpair(p1)
%
I=[];
for j=1:1:length(p2)
for i=1:1:length(p1)
     if (abs(p1(i)-p2(j)) < 0.0001)
        I=[I,i];
         end
    end
end
I=I';
```

【例 7.21】 直接型结构到格型梯形结构转换

已知 IIR 滤波器系统函数 $H(z)$

$$H(z)=\frac{1+0.8z^{-1}-z^{-2}-0.8z^{-3}}{1-1.7z^{-1}+1.53z^{-2}-0.648z^{-3}}$$

将 $H(z)$转换成格型梯形结构（零-极点系统的 Latice 结构，见文献[17]）。

解：■ 建模

格型结构是一种很有用的结构，在功率谱估计、语音处理和自适应滤波等方面已得到广泛应用。FIR 和全极点 IIR 系统可用格型结构实现，而零-极点 IIR 系统要用“格型梯形结构”实现。从文献[4]可知，由直接型系统函数 $H(z)$计算或递推格型结构（或格型梯形结构）系数很麻烦。MATLAB 信号处理工具箱函数 tf2latc 和 latc2tf 使直接型到格型结构和格型到直接型结构的转换非常容易。下面结合本例和例 7.22、例 7.23 说明这两个函数的功能与使用格式。

● tf2latc 函数实现直接型到格型转换

➢ [K, C] = tf2latc (B, A)　　求出零-极点 IIR 系统格型梯形结构的格型参数向量 K 和梯形参数向量 C（用 A(1)归一化）。注意，当系统函数在单位圆上有极点时发生错误。

➢ K = tf2latc (1, A)　　求出全极点 IIR 系统的格型结构参数向量 K。如果使用格式[K, C] = tf2latc (1, A)，返回的系数 C 为标量。

➢ K = tf2latc (B)　　求出 FIR 系统的格型梯形结构参数（反射系数）向量 K（用 H(z)的常数项 B(1)归一化）。

系数向量 K 和 C 与格型结构的对应关系见本例和例 7.22、例 7.23 的程序运行结果及其格型结构图。

函数 latc2tf 实现与 tf2latc 相反的结构转换。其使用格式可用 help 命令查到。

应当注意，线性相位 FIR 滤波器不能用格型结构实现。

■ MATLAB程序q721.m

```
% 零-极点 IIR 系统的直接型结构到格型梯形结构转换
B=[1, 0.8, -1, -0.8];
A=[1, -1.7, 1.53, -0.648];
[K, C]=tf2latc(B, A)
```

■ 程序运行结果：

```
K =  -0.7026
      0.7385
     -0.6480
C =   1.6212
     -0.8586
     -2.3600
     -0.8000
```

格型梯形网络结构如图 7.21 所示。

【例 7.22】 FIR 滤波器直接型到级联型和格型转换

FIR 滤波器系统函数为

$$H(z)=2+\frac{13}{12}z^{-1}+\frac{5}{4}z^{-2}+\frac{2}{3}z^{-3}$$

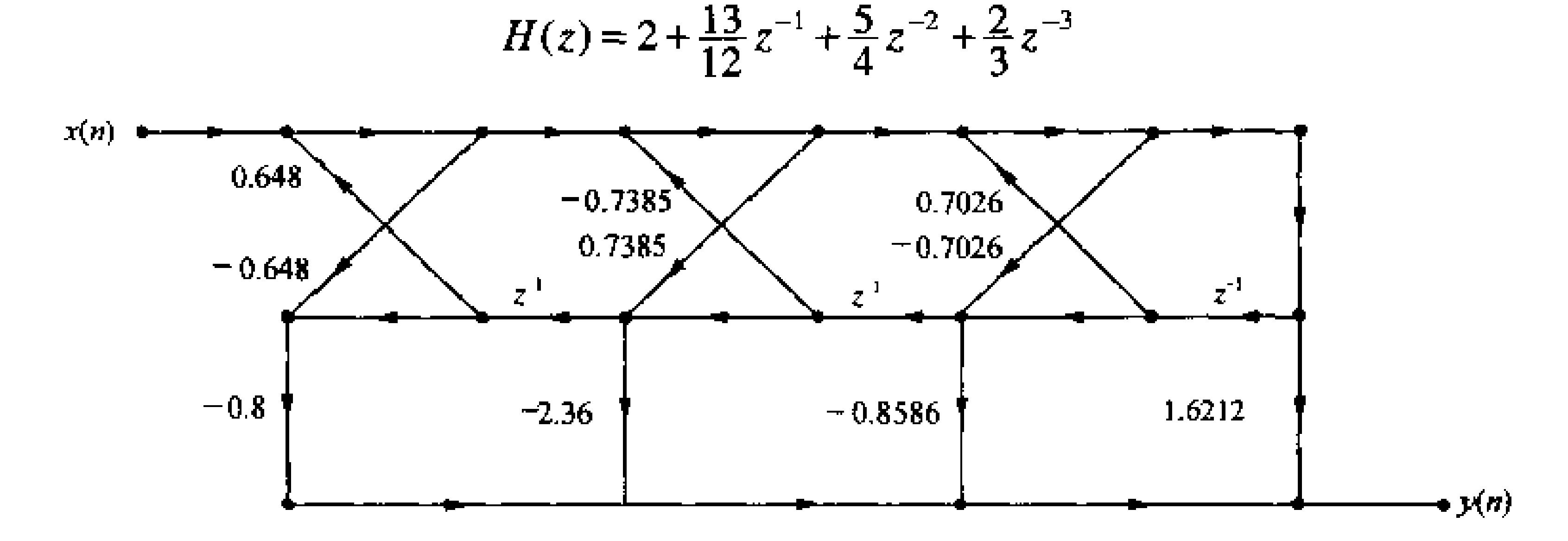


图 7.21　格型梯形结构

求出其级联型结构和格型结构。

解： 调用信号处理工具箱函数 tf2sos 和 tf2latc 使本题求解程序 q722.m 非常简洁。

■ **MATLAB程序q722.m**

```
% FIR 直接型到级联型和格型转换
clear;
b=[2,13/12,5/4,2/3]; a=1;          % 设定参数
fprintf('级联型结构系数:');
[sos,g]=tf2sos(b,a)                %直接型到级联型转换
fprintf('格型结构系数（反射系数）: ');
[K]=tf2latc(b)                     %直接型到格型转换
```

■ **程序运行结果**

级联型结构系数

```
sos = 1.0000    0.5360         0        1.0000        0        0
      1.0000    0.0057    0.6219        1.0000        0        0
g =     2
```

格型结构系数（反射系数）:

```
K =  0.2500    0.5000    0.3333
```

由级联型结构系数写出 $H(z)$表达式。

$$H(z)=2(1+0.536z^{-1})(1+0.0057z^{-1}+0.6219z^{-2})$$

级联型结构如图 7.22-1 所示。

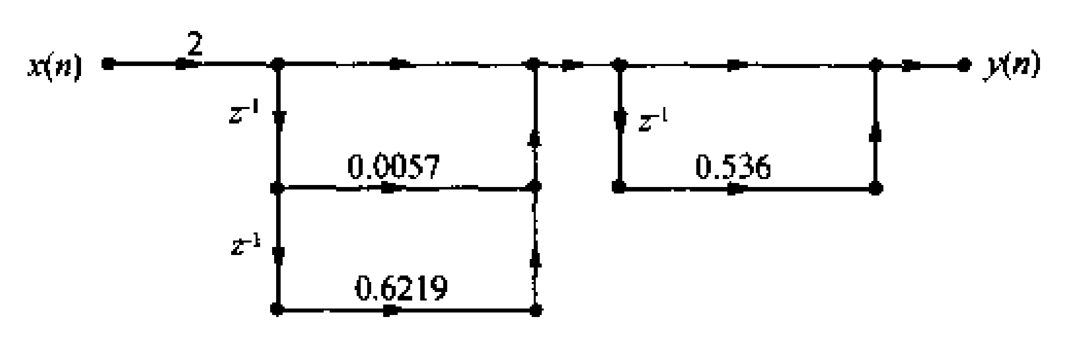


图 7.22-1　级联型结构

格型结构如图 7.22-2 所示。应当注意，由于函数 tf2latc 所求的格型结构是用 $H(z)$的常数项 b_0 归一化的，所以，结构图中要乘以 $b_0=2$ 才能保证原滤波器增益不变。

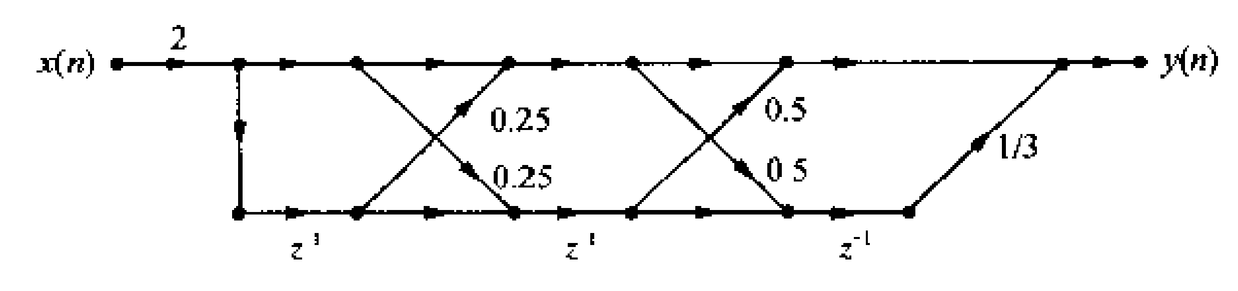


图 7.22-2　格型结构

【例 7.23】 FIR 格型结构到直接型结构转换

已知 FIR 滤波器的格型结构如图 7.23-1 所示。

求其系统函数 $H(z)$，画出直接 II 型结构图。

解： 由图 7.23-1 可得到反射系数向量为

$\boldsymbol{K}=[2,\ 1/4,\ 1/2,\ 1/3]$

调用函数 latc2tf，求解本题的程序 q723.m 如下。

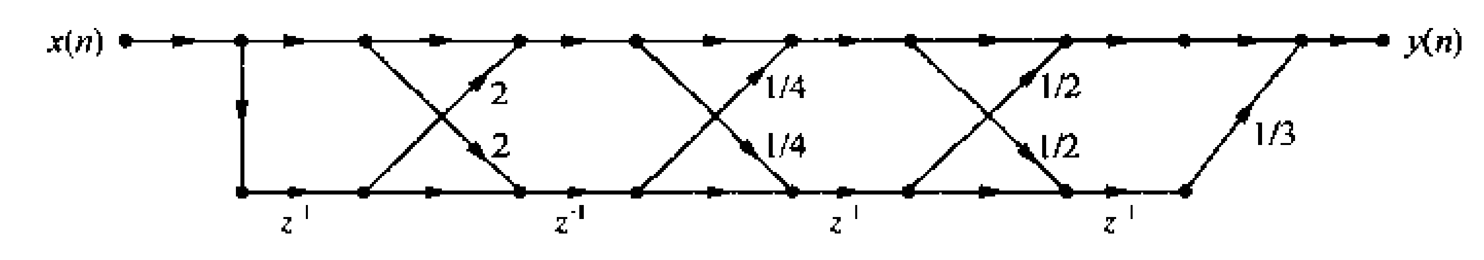


图 7.23-1　FIR 格型网络结构

■ **MATLAB程序q723.m**

```
% 格型结构到直接型结构转换
clear;
K=[2,1/4,1/2,1/3];
fprintf('直接型结构系数:');
B=latc2tf(K) %格型到直接型转换
```

■ **程序运行结果**

直接型结构系数

```
B = 1.0000    2.7917    2.0000    1.3750    0.3333
```

由直接型结构系数向量 $\boldsymbol{B}$ 写出系统函数 $H(z)$的表达式。

$$H(z)=1+2.7917z^{-1}+2z^{-2}+1.375z^{-3}+0.3333z^{-4}$$

直接型结构如图 7.23-2 所示。

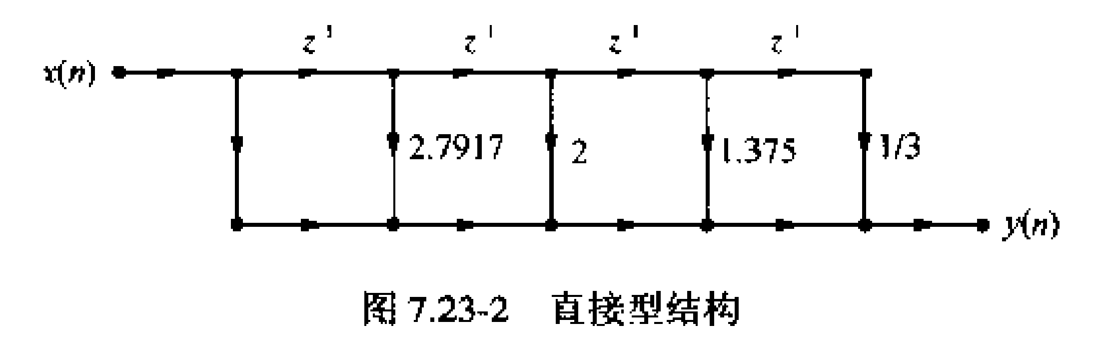


图 7.23-2　直接型结构

7.5　FIR 数字滤波器设计

先简单介绍一下数字滤波器的类型和设计指标。

数字滤波器可分为 FIR（有限脉冲响应）和 IIR（无限脉冲响应）两种。IIR 滤波器的系统函数是两个 z 的多项式的有理分式，而 FIR 滤波器的分母为 1，即只有一个分子多项式。

数字滤波器的理想幅频特性如图 7.24 所示（为方便起见，以后图中的标注均采用程序字体）。在 0 到π的全部频段上，其幅值为 1 的区域为通带。其余为阻带，即其幅值为 0。根据 wc1 和 wc2 取值不同可分为 4 种类型：

（1）低通滤波器，当 wc1=0 时；

（2）高通滤波器，当 wc2 = π 时；

（3）带通滤波器，当 wc1 及 wc2 如图 7.24 所示时；

（4）带阻滤波器，当[0，wc1]及[wc2，1]区间幅度为 1，[wc1，wc2] 区间幅度为 0 时。

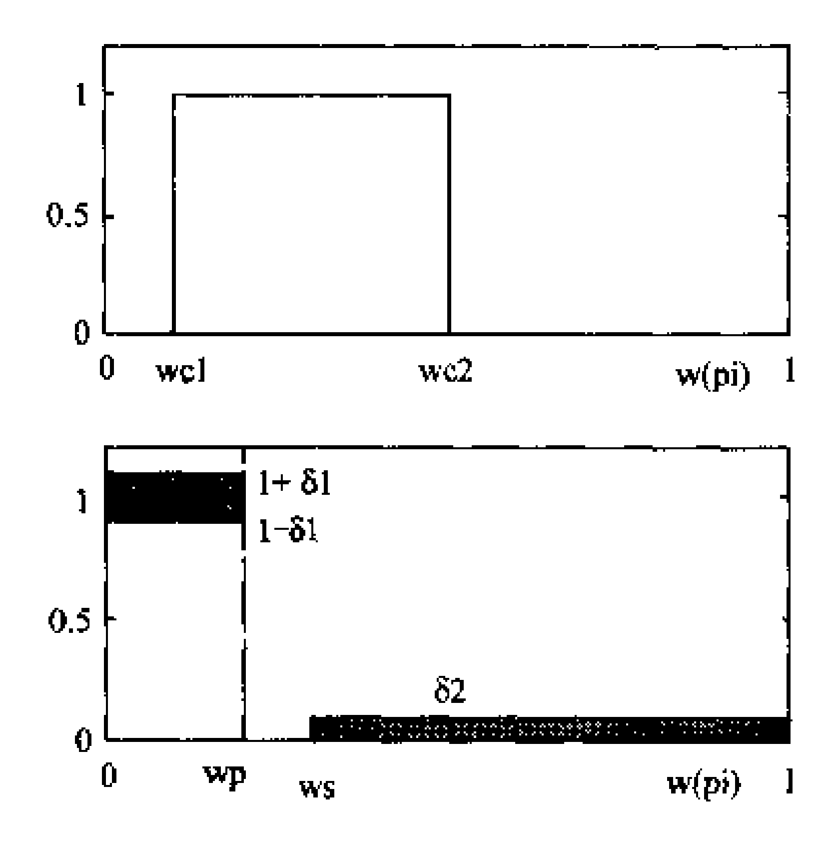


图 7.24　滤波器的特性和指标

有些情况下，还对滤波器的相位特性提出要求，理想的是线性相位特性，即相移与频率成线性关系。

实际的滤波器不可能完全实现理想幅频特性，必有一定误差，因此要规定适当的指标。

以低通滤波器为例，在[0，wp]的通带区，幅频特性会在 1 附近波动±δ1；在 ws～1 的阻带区，幅频特性不会真等于零，是一个大于零的δ2 值；wp 也不可能等于 ws，在[wp，ws]之间，为过渡区；这三个与理想特性的不同点，就构成了滤波器的指标体系。即通带频率 wp 和通带波动δ1，阻带频率 ws 和阻带衰减δ2。

在许多情况下，人们习惯用分贝为单位，定义通带波动为 R_p（分贝），阻带衰减为 R_s（分贝）。

$$R_p = -20\log_{10}\left(\frac{1-\delta_1}{1+\delta_1}\right) > 0, \quad R_s = -20\log_{10}\left(\frac{\delta_2}{1+\delta_1}\right) > 0$$

对于带通滤波器，wp 应表为[wp1，wp2]；对于带阻滤波器，ws 应表为[ws1，ws2]。其他复杂形状的预期特性通常也可由若干理想的幅频特性叠合构成。

FIR 数字滤波器最大的优点是容易设计成线性相位特性，而且不存在稳定性问题。线性相位特性滤波器在图像处理和数字通信等领域非常有用。MATLAB 信号处理工具箱中提供的 FIR 数字滤波器的设计方法有以下两种。

（1）窗函数法

用窗函数法设计 FIR 数字滤波器时，先根据ω_c 和 N 求出相应的理想滤波器单位脉冲响应 $h_d(n)$。

$$H_d(e^{j\varpi})=\begin{cases}e^{j\varpi\alpha} & 0\leqslant|\varpi|\leqslant\omega_c\\ 0 & \omega_c<|\varpi|\leqslant\pi\end{cases}$$

$$h_d(n)=\frac{1}{2\pi}\int_{-\omega_c}^{\omega_c}H_d(e^{j\varpi})e^{j\varpi n}d\varpi=\frac{\sin(\varpi_c(n-\alpha))}{\pi(n-\alpha)}$$

其中α是一个常数时刻。这样由 MATLAB 得出的 hd(n)是一个无限长序列，因而不可用 FIR 滤波器实现。所以第二步要选择合适的窗函数 w(n)来截取 hd(n)的适当长度（即阶数），以保证实现要求的阻带衰减；最后得到 FIR 滤波器单位脉冲响应 h(n)=hd(n)w(n)。

（2）等波纹最佳一致逼近法（也称 Parks-McClellan 最优法）

信号处理工具箱采用 remez 算法实现线性相位 FIR 数字滤波器的等波纹最佳一致逼近设计。与其他设计法相比，其优点是，设计指标相同时，使滤波器阶数最低；或阶数相同时，使通带最平坦，阻带最小衰减最大；通带和阻带均为等波纹形式，最适合设计片段常数特性的滤波器。其调用格式如下：

➢ b = remez(N, f, m, w, 'ftype')

其中，w 和 ftype 可缺省。b 为滤波器系数向量，调用参数 N，f，m 的含义与函数 fir2 中类同，但这里有一点不同，期望逼近的幅频响应值位于 f(k)与 f(k+1)（k 为奇数）之间的频段上（即（f(k)，m(k)）与（f(k+1)，m(k+1)）两点间的连线），而 f(k+1)与 f(k+2)之间为无关区。w 为加权向量，其长度为 f 的一半。w(k)为对 m 中第 k（k 为奇数）个常数片段的逼近精度加权值，w 值越大逼近精度越高。ftype 用于指定滤波器类型，可用 help 命令查看。

应当注意，f 中不能出现重复频点，即 remez 函数不能逼近理想频响特性。

remezord 函数用于估算 FIR 数字滤波器的等波纹最佳一致逼近设计的最低阶数 N，从而使滤波器在满足指标的前提下造价最低。基本调用格式如下：

➢ [N, fo, mo, w] = remezord(f, m, dev, Fs)

其返回参数供 remez 函数使用。设计的滤波器可以满足由参数 f，m，dev 和 Fs 指定的指标。f 和 m 与 remez 中所用的类似，这里 f 可以是模拟频率（Hz）或归一化数字频率，但必须以 0 开始，以 Fs/2（用归一化频率时为 1）结束，而且其中省略了 0 和 Fs/2 两个频点。Fs 为采样频率，省略时默认为 2Hz。

dev 为各逼近频段允许的幅频响应偏差（波纹振幅）。

remez 函数可直接调用 remezord 返回的参数，使用格式如下：

➢ b=remez(N, fo, mo, w)

本节将举例说明这两种设计方法的 MATLAB 编程方法。

【例 7.24】 用各种窗函数设计 FIR 数字滤波器

分别用矩形窗和 Hamming 窗设计线性相位 FIR 低通滤波器。要求通带截止频率$\omega_c=\pi/4$，单位脉冲响应 $h(n)$的长度 $N=21$。绘出 $h(n)$及其幅频响应特性曲线。

解：■ 建模

用窗函数法设计 FIR 数字滤波器时，先求出相应的理想滤波器（本例应为理想低通）单位脉冲响应 $h_d(n)$，再根据阻带最小衰减选择合适的窗函数 $w(n)$，最后得到 FIR 滤波器单位脉冲响应 $h(n)=h_d(n)w(n)$。

本题中，$\omega_c=\pi/4$，$N=21$，所以线性相位理想低通滤波器的单位脉冲响应为

$$h_d(n)=\frac{\sin\left(\dfrac{\pi(n-\alpha)}{4}\right)}{\pi(n-\alpha)}$$

为了满足线性相位 FIR 滤波器条件 $h(n)=h(N-1-n)$，要求$\alpha=(N-1)/2=10$。

信号处理工具箱中有窗生成函数 boxcar，hamming，hanning 和 blackman 等。wn=boxcar (m) 产生长度为 m 的矩形窗函数列向量 wn。其他窗函数产生工具箱函数的调用格式相同。

■ **MATLAB程序q724.m**

```
%用窗函数法设计 FIR 低通滤波器`
clear;close all
N=21; wc=pi/4;                          % 理想低通滤波器参数
n=0:N-1; r=(N-1)/2;
hdn=sin(wc* (n-r))/pi./(n-r);           % 计算理想低通单位脉冲响应 h_d(n)
if rem(N,2)~=0    hdn(r+1)=wc/pi; end %N 为奇数时,处理 n=r 点的 0/0 型
wn1=boxcar(N);           % 矩形窗
hn1=hdn.*wn1';           % 加窗
% 以上两条语句可代以 fir 函数: hn1=fir1(N-1,wc/pi,boxcar(N));
wn2=hamming(N);          % hamming 窗
hn2=hdn.*wn2';           % 加窗
% 以上两条语句可代以 fir 函数: hn2=fir1(N-1,wc/pi,hamming(N));k=3;
……
```

用 stem(n，hn，'.')，hw=fft(hn，512)，w=2* [0:511]/512，plot(w，20*log10(abs(hw))) 语句分别画出此 FIR 滤波器的系数序列（即其脉冲响应）及幅频特性。图形划分及标注语句略。

■ **程序运行结果**

对两种窗函数的设计结果分别如图 7.25-1 和图 7.25-2 所示。由图中可以看出，不同的窗函数生成的过渡带宽度和阻带最小衰减是不同的。这就是选择窗函数的根据。

MATLAB 信号处理工具箱提供了基于窗函数法的 FIR 滤波器的设计函数 fir1 和 fir2，它们使本题的设计程序更加简单。下面简要介绍 fir1 和 fir2 的功能、格式及使用说明。

● fir1

功能：基于窗函数的 FIR 滤波器设计——标准频率响应形状。

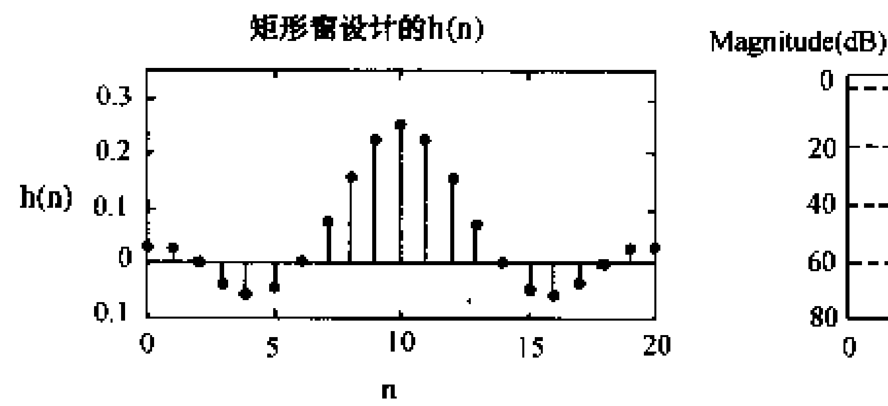
格式：b=fir1(N, wc, 'ftype', window)。

说明：标准频率响应指所设计的滤波器的预期特性为理想频率响应，包括低通、带通、高通或带阻特性。

ftype 和 window 可以缺省。b=fir1(N，wc)可得到截止频率为 wc 且满足线性相位条件的 N 阶 FIR 低通滤波器，window 默认选用 hamming 窗。其单位脉冲响应 $h(n)$为

$$h(n)=b(n+1)\qquad n=0,\ 1,\ 2,\ \cdots,\ N$$

见程序 q724.m 中的语句。当 wc=[wc1，wc2]时，得到的是通带为 wc1≤w≤wc2 的带通滤波器。



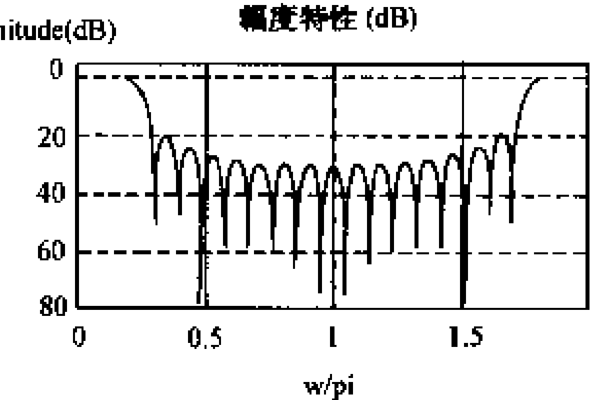
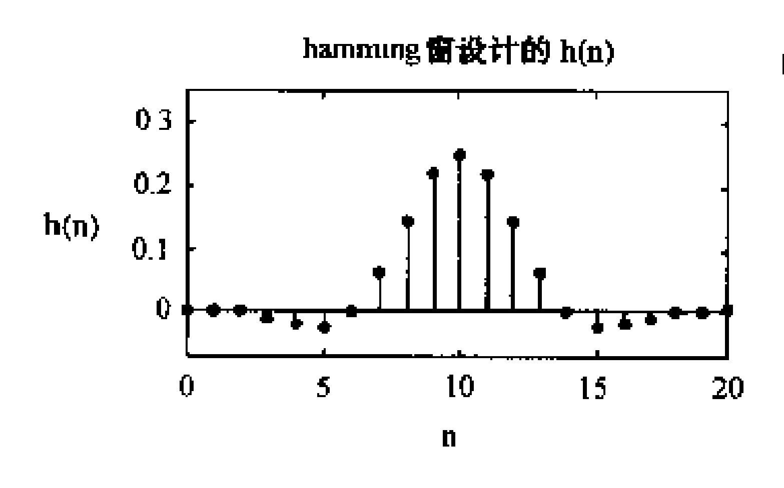


图 7.25-1　矩形窗设计结果



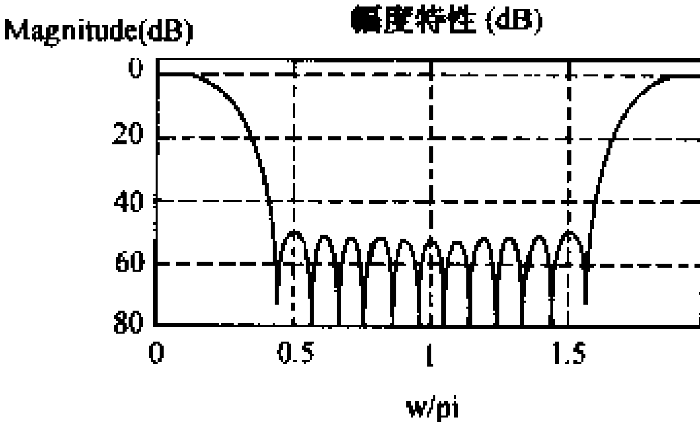


图 7.25-2　hamming 窗设计结果

➢ b = fir1(N, wc, 'ftype')　　可设计高通和带阻滤波器。

• 当 ftype=high 时，设计高通 FIR 滤波器；

• 当 ftype=stop 时，设计带阻 FIR 滤波器。

应当注意，在设计高通和带阻滤波器时，阶数 N 只能取偶数（h(n)的长度 N+1 为奇数）。不过，当用户将 N 设置为奇数时，fir1 会自动对 N 加 1。

window 默认为 hamming 窗。可用的其他窗函数有 boxcar，hanning，bartlett，blackman，kaiser 和 chebwin 窗。这些窗函数的使用很简单（可用 help 命令查到），例如：

➢ b = fir1(N, wc, bartlett(N+1))　　使用 bartlett 窗设计。

● fir2

功能：基于窗函数的 FIR 滤波器设计——任意频率响应形状。

格式：b = fir2(N, f, m, window)

说明：fir2 函数用于设计具有任意频率响应形状的加窗线性相位 FIR 数字滤波器，其幅频特性由频率点向量 f 和幅度值向量 m 给出，$0 \leqslant f \leqslant 1$，要求 f 为单增向量，而且从 0 开始，以 1 结束，1 表示数字频率 $w = \pi$。m 与 f 等长度，m(k)表示频点 f(k)的幅频响应值。plot(f, m)命令可画出期望逼近的幅频响应曲线。N 和 window 与 fir1 中的相同。

例如，调用 fir2 函数逼近截止频率 $wc = 0.6\pi$的理想高通的 30 阶 FIR 数字滤波器设计程序如下。

■ MATLAB程序q724a.m

```
%fir2 使用举例
clear;close all
f=[0,0.6,0.6,1];m=[0,0,1,1];              % 预期设定幅频响应
b=fir2(30,f,m);n=0:30;                    % 设计 FIR 数字滤波器系数
subplot(3,2,1);stem(n,b,'.')
```

```
xlabel('n');ylabel('h(n)');
axis([0,30,-0.4,0.5]),line([0,30],[0,0])
[h,w]=freqz(b,1,256);
subplot(3,2,2);plot(w/pi,20*log10(abs(h)));grid
axis([0,1,-80,0]);xlabel('w/pi');ylabel('幅度（dB）');
```

■ 程序运行结果如图 7.26 所示

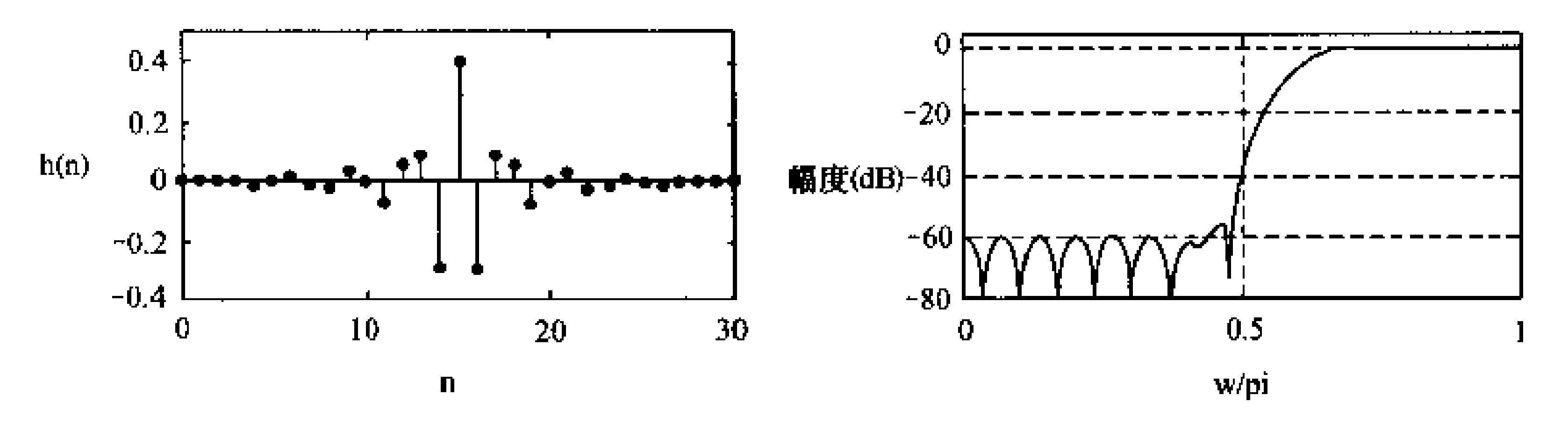


图 7.26　一个理想高通的 30 阶 FIR 数字滤波器设计结果

【例 7.25】 用窗函数法设计 FIR 带通滤波器

用窗函数法设计一个 FIR 带通滤波器。指标如下：

低端阻带截止频率　$\omega_{ls}=0.2\pi$；

低端通带截止频率　$\omega_{lp}=0.35\pi$；

高端通带截止频率　$\omega_{up}=0.65\pi$；

高端阻带截止频率　$\omega_{us}=0.8\pi$；

通带最大衰减　$R_p=1$ dB；

阻带最小衰减　$R_s=60$ dB。

绘出 $h(n)$及其幅频特性曲线。

解：■ 建模

设计原理与例 7.24 相同。关键在于如何选定 N，才能满足题目给出的要求。为了简化程序，使用工具箱函数 fir1 的格式 b = fir1(N, wc, window)来编写程序。

由于设计的是带通滤波器，所以参数ω_c为行向量$\omega_c=[\omega_{lp}/\pi，\omega_{hp}/\pi]$

根据阻带最小衰减 R_s = 60dB 选择窗函数类型和阶次。有两个方法：①查窗函数基本参数表；②在命令窗中键入 sptool 函数，单击 new design 及 edit design 后在图形界面上选择类型和阶次，直至满足要求。选 blackman 窗，其滤波器阻带最小衰减可达到 74dB，其窗口长度 M 由过渡带宽度 B = 0.15π 决定，lackman 窗设计的滤波器过渡带宽度为 $12\pi/M$，故 M 取 80。因 $M=N+1$，所以滤波器阶数 $N=79$。

■ MATLAB程序q725.m

```
% 用窗函数法设计 FIR 带通滤波器
clear;close all;
wls=0.2*pi;wlp=0.35*pi;whp=0.65*pi;
B=wlp-wls;                    %计算过渡带宽
N=ceil(12/0.15);          %计算窗口长度
wc=[wlp/pi-6/N,whp/pi+6/N];      % 设置理想带通截止频率
```

```
hn=fir1(N-1,wc,blackman(N));   % 设计滤波器系数
......
```

以下语句绘出脉冲响应及幅频特性图，绘图及标注语句略。

■ **程序运行结果**

如图 7.27 所示。完全满足指标要求。请读者思考程序中设置 wc 向量的原理。

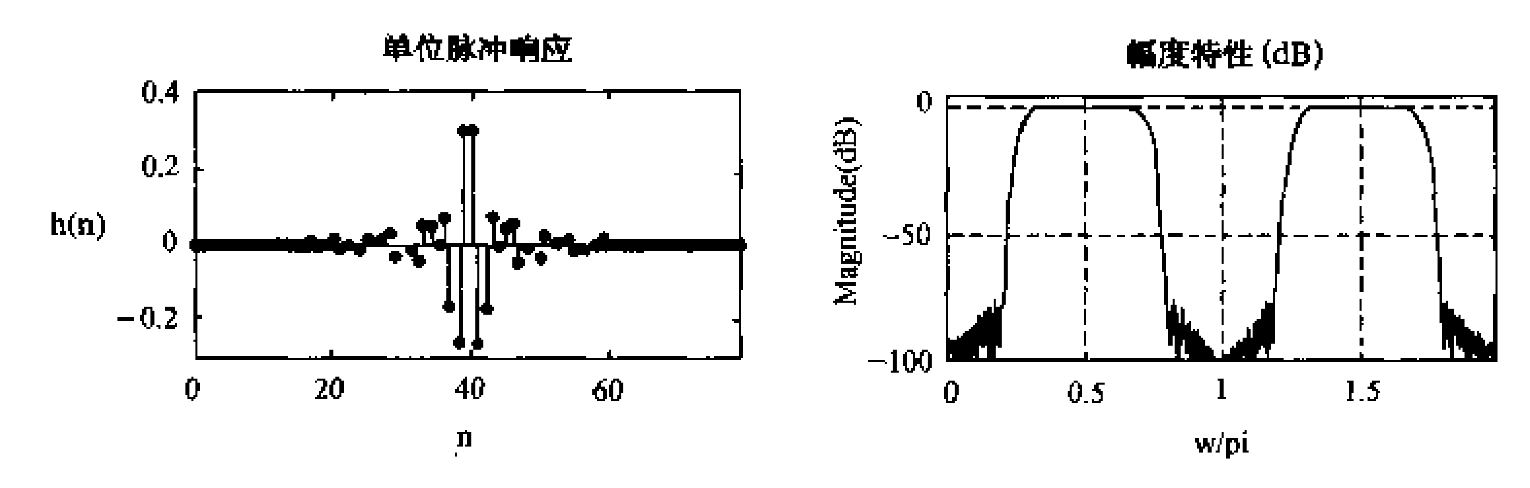


图 7.27　用窗函数法设计的 FIR 带通滤波器 $h(n)$及其幅度特性

【例 7.26】 用 remez 函数设计 FIR 低通滤波器

设计滤波器，使逼近低通滤波特性$|H_d(e^{j\omega})|$。

$$|H_d(e^{j\omega})|=\begin{cases}1 & 0\leqslant|\varpi|\leqslant\pi/4\\0 & 5\pi/16\leqslant|\varpi|\leqslant\pi\end{cases}$$

要求通带波纹$\alpha_p\leqslant3$dB，阻带衰减$\alpha_s\geqslant60$dB，并用最小阶数实现。

绘出设计的 FIR 数字滤波幅频特性曲线。检验设计指标。

解：■ 建模

先由题意计算设计参数 $f=[1/4,\ 5/16]$，$m=[1,\ 0]$;

dev 的计算稍复杂一些，由于

$$R_p=20\log\left(\frac{1+\text{dev}(1)}{1-\text{dev}(1)}\right),\quad A_s=-20\log(\text{dev}(2))$$

所以　　$\text{dev}(1)=(10^{R_p/20}-1)/(10^{R_p/20}+1)$，$\text{dev}(2)=10^{(-A_s/20)}$

有了这几个参数就可以调用 remezord 和 remez 函数了，于是可得如下程序。

■ **MATLAB程序q726.m**

```
%用 remez 函数设计 FIR 低通滤波器
clear;close all
fc=1/4; fs=5/16;                 % 输入给定指标
Rp=3; As=60; Fs=2;
f=[fc,fs];m=[1,0];          % 计算 remezord 函数所需参数 f,m,dev
dev=[(10^(Rp/20)-1)/(10^(Rp/20)+1),10^(-As/20)];
[N,fo,mo,W]=remezord(f,m,dev,Fs);% 确定 remez 函数所需参数
hn=remez(N,fo,mo,W);             % 调用 remez 函数进行设计
hw=fft(hn,512);                  % 求设计出的滤波器频率特性
w=[0:511] *2/512;
plot(w,20*log10(abs(hw)));grid;   % 画对数幅频特性图
axis([0,max(w)/2,-90,5]);
```

```
xlabel('w/pi');ylabel('Magnitude(dB)')
line([0,0.4],[-3,-3]);          %画线检验设计结果
line([1/4,1/4],[-90,5]);line([5/16,5/16],[-90,5]);
```

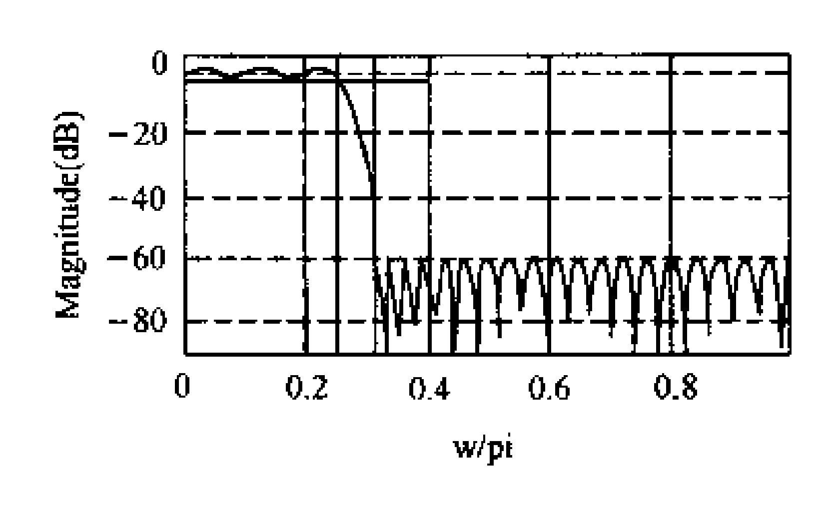


图 7.28　程序输出的幅频特性

■ 程序运行结果

如图 7.28 所示。图中，横线为-3dB，两条竖线分别位于频率π/4 和 5π/16。显然，通带指标稍有富裕，过渡带宽度和阻带最小衰减刚好满足指标要求。

【例 7.27】 用 remez 函数设计高通滤波器

观察等波纹逼近法中加权系数 $w(\omega)$及滤波器阶数 N 的作用和影响。期望逼近的滤波器通带为[3π/4，π]，阻带为[0，23π/32]。

解：■ 建模

在滤波器设计中，技术指标越高，实现滤波器的阶数也就越高。另外，对固定的阶数，通带与阻带指标可以互换，过渡带宽度与通带波纹和阻带衰减指标可以互换。用等波纹最佳一致逼近设计函数 remez 很容易验证以上设计原则。

在 remez 函数调用格式 b = remez(N, f, m, w)中，f = [0, 3/4, 23/32, 1]，m = [0, 0, 1, 1]。其余参数分三种情况进行设计，①N = 30，w = [1, 1]；②N = 30，w = [1, 5] ③N = 60，w = [1, 1]。

■ MATLAB程序q727.m

```
% 用 remez 函数设计 FIR 数字高通滤波器
clear; close all
f=[0, 23/32, 3/4, 1];m=[0, 0, 1, 1];
N1=30;W1=[1, 1]; hn1=remez(N1, f, m, W1);        % 情况①
k=[0:1023]*2/1024;
Hw1=fft(hn1, 1024); plot(k, 20*log10(abs(Hw1))); %求出其幅频特性
axis([0,1,-40,5]),grid on,pause             % 只画出正半轴频谱
N2=30;W2=[1, 5]; hn2=remez(N2, f, m, W2);        % 情况②
Hw2=fft(hn2, 1024); plot(k, 20*log10(abs(Hw2)));%求出其幅频特性
axis([0,1,-40,5]),grid on,pause             % 只画出正半轴频谱
N3=60;W3=[1, 1]; hn3=remez(N3, f, m, W3);        % 情况③
Hw3=fft(hn3, 1024); plot(k, 20*log10(abs(Hw3)));%求出其幅频特性
axis([0,1,-40,5]),grid on,pause             % 只画出正半轴频谱
……
```

图形划分及标注语句从略。

■ 程序运行结果

如图 7.29（a），（b）和（c）所示。

由图可见，w 较大的频段逼近精度较高；w 较小的频段逼近精度较低。N 较大时逼近精度较高；N 较小时逼近精度较低。

本节所有例题只绘出了 FIR 滤波器的单位脉冲响应 $h(n)$及其幅频响应特性曲线，这主要是为了直观。但是，实际设计滤波器时，需要得到系数 $h(n)$的值。因为

$$h(n) = hn(n+1),\ n = 0,\ 1,\ \cdots,\ N$$

程序运行后，在 MATLAB 命令窗口键入 hn 可得到系数 $h(n)$的值。

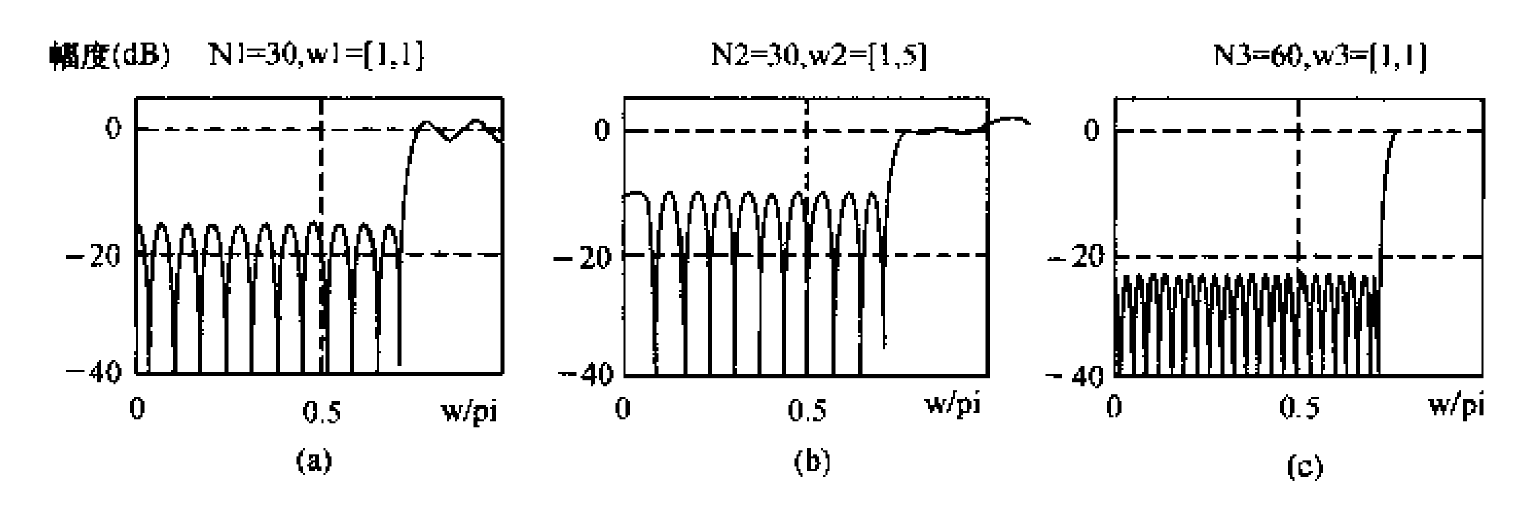


图 7.29　程序 q727.m 运行结果

7.6　IIR 数字滤波器设计

IIR 数字滤波器设计的主要方法是先设计低通模拟滤波器，进行频率变换，将其转换为相应的（高通、带通等）模拟滤波器，再转换为高通、带通或带阻数字滤波器。对设计的全过程的各个步骤，MATLAB 都提供了相应的工具箱函数，使 IIR 数字滤波器设计变得非常简单。本节主要结合例题介绍这些 IIR 滤波器设计的工具箱函数。

IIR 数字滤波器的设计步骤可用图 7.30 所示的流程图来表示：

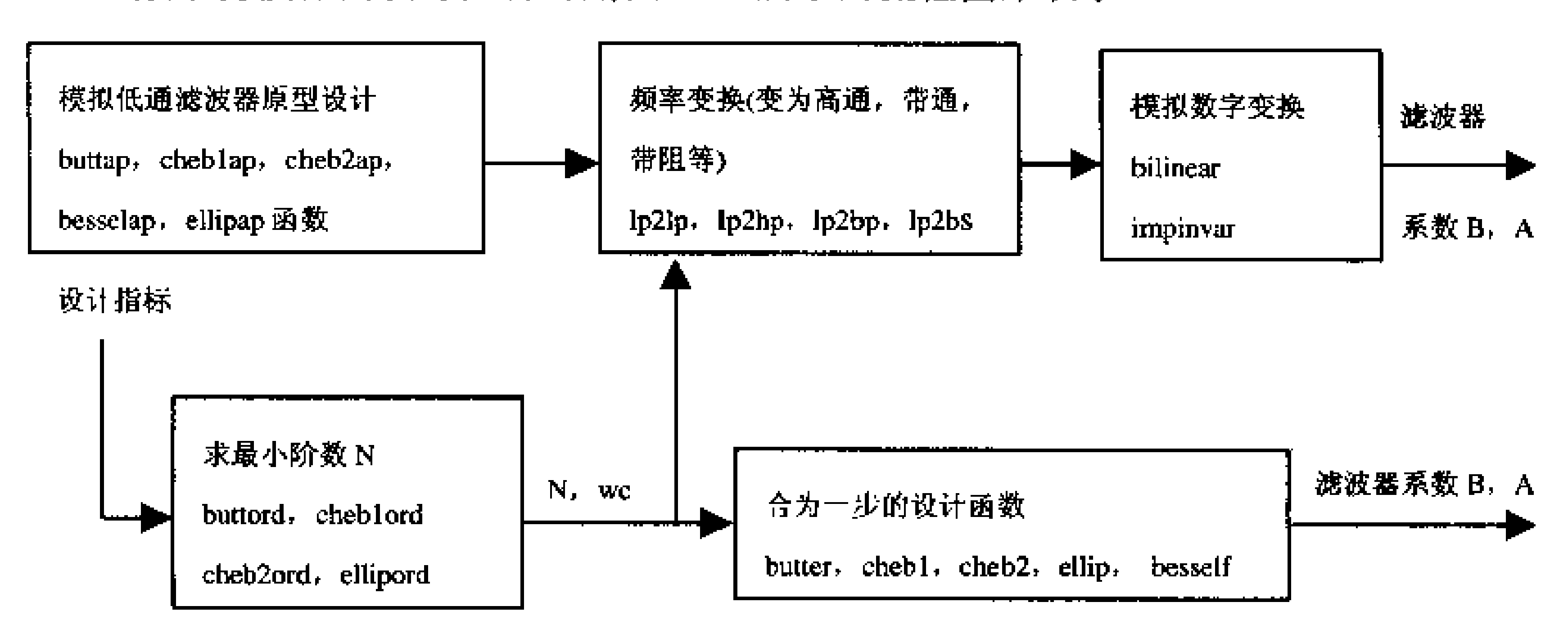


图 7.30　IIR 数字滤波器设计步骤流程图

这个图也清楚地表明了 5 类 20 个信号处理工具箱函数的作用。下面以巴特沃斯滤波器设计函数为典型，介绍此流程图中函数的功能和用法。其他类型的滤波器设计函数用法可类推。

（1）求最小阶数 N 的函数 buttord

➢ [N, wc] = buttord (wp, ws, Rp, Rs, 's')　　根据滤波器指标 wp，ws，Rp，Rs，求出巴特沃斯模拟滤波器的阶数 N 及频率参数 wc，此处 wp，ws 及 wc 均以弧度/秒为单位。

去掉最后变元 s 后，它就用于数字滤波器。

➢ [N, wc] = buttord (wp, ws, Rp, Rs)　　求出巴特沃斯数字滤波器的阶数 N 及频率参数 wc，此处 wp，ws 及 wc 均在[0，1]区间归一化，以π 弧度为单位。对带通或带阻滤波器，wp，ws 都是两元变量。如

带通滤波器:　wp = [.2　.7]，ws = [.1　.8]

带阻滤波器:　wp = [.1　.8]，ws = [.2　.7]

（2）模拟低通滤波器原型设计函数 buttap

➢ [z，p，k] = buttap(N)　　得到[z, p, k]后，很容易求出滤波器系数 B，A。

（3）模拟频率变换函数 lp2lp

➢ [Bt, At] = lp2lp(B, A, wo)　　把单位截止频率的模拟低通滤波器系数 B，A 变为另一截止频率 wo[弧度/秒]的低通滤波器系数 Bt，At。参阅例 7.32。

（4）模拟数字变换函数——双线性变换函数 bilinear 或脉冲响应不变法函数 impinvar

➢ [Bd, Ad] = bilinear (B, A, Fs)　　把模拟滤波器系数 B，A 变为近似等价的数字滤波器系数 Bd，Ad。详细用法见例 7.29。

（5）把（2）、（3）、（4）合为一步的数字滤波器设计函数 butter(N, wc, 'ftype')

➢ [B, A] = butter (N, wc)　　设计低通或带通数字滤波器系数 B，A(当为带通滤波器时，第(1)类函数由 wp = [wp1, wp2]会自动生成 wc = [w1, w2])。

➢ [B, A] = butter (N, wc, 'high')　　设计高通数字滤波器系数 B，A。

➢ [B, A] = butter (N, wc, 'stop')　　设计带阻数字滤波器系数 B，A。

butter(N, wc, 'ftype')还有零极增益和状态空间形式，读者可用 help 命令查阅。

由此可见，通常情况下，有了（1）和（5）两类函数，数字滤波器设计问题也就解决了。注意（5）中采用的是双线性变换函数 bilinear（如要用脉冲响应不变法就得分步来做）。

由于模拟滤波器和 IIR 数字滤波器系统函数的系数向量在 MATLAB 中表示的意义有区别，为了避免混淆，这里做一些说明。

① *N* 阶模拟滤波器系统函数的一般形式

$$H_a(s)=\frac{b_N s^N+b_{N-1}s^{N-1}+\cdots+b_1 s+b_0}{a_N s^N+a_{N-1}s^{N-1}+\cdots+a_1 s+a_0}=\frac{B(s)}{A(s)} \tag{7.14}$$

② *N*阶数字滤波器系统函数的一般形式

$$H(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+\ldots+b_{N-1}z^{-(N-1)}+b_Nz^{-N}}{1+a_1z^{-1}+a_2z^{-2}+\ldots+a_{N-1}z^{-(N-1)}+a_Nz^{-N}}=\frac{B(z)}{A(z)} \tag{7.15}$$

③ MATLAB中，$H_a(s)$和$H(z)$的表示

由上述可见，$H_a(s)$完全由分子多项式系数$\{b_k\}$和分母多项式系数$\{a_k\}$决定，标准形式a_0=1。所以 MATLAB 中，一般用 N+1 维行向量 $\boldsymbol{B}$ 来表示分子多项式 $B(s)$或 $B(z)$的 N+1 个系数$\{b_k\}$；用 N+1 维行向量 $\boldsymbol{A}$ 来表示分母多项式 $A(s)$或 $A(z)$的 N+1 个系数$\{a_k\}$。换言之，当用 MATLAB 设计的程序运行结束后，得到系数向量 $\boldsymbol{B}$ 和 $\boldsymbol{A}$。

对模拟滤波器设计程序，相应的系统函数为

$$H_a(s)=\frac{B(s)}{A(s)}=\frac{B(1)s^N+B(2)s^{N-1}+\ldots+B(N)s+B(N+1)}{A(1)s^N+A(2)s^{N-1}+\ldots+A(N)s+A(N+1)} \tag{7.16}$$

即系数关系为

$$b_k=B(N+1-k),\ a_k=A(N+1-k),\qquad k=0,\ 1,\ 2,\ \cdots,\ N$$

对数字滤波器设计程序，相应的系统函数为

$$H(z)=\frac{B(z)}{A(z)}=\frac{B(1)+B(2)z^{-1}+\ldots+B(N)z^{-(N-1)}+B(N+1)z^{-N}}{A(1)+A(2)z^{-1}+\ldots+A(N)z^{-(N-1)}+A(N+1)z^{-N}} \tag{7.17}$$

即 $b_k=B(k+1)$，$a_k=A(k+1)$， $k=0, 1, 2, \cdots, N$

在工程实际中，需要的设计结果是系数向量 **B** 和 **A**，用 **B** 和 **A** 来综合滤波器的硬件实现结构或软件运算结构。为了直观地看出设计效果，本节的例题均以滤波器幅频响应曲线作为设计结果输出。如果用户需要滤波器系数，在程序运行后，只要在 MATLAB 命令窗口键入系数向量名，则相应的系数就显示出来（见例 7.30 和例 7.31）。

【例 7.28】 低通巴特沃斯模拟滤波器设计

设计一个低通巴特沃斯模拟滤波器，指标如下。

通带截止频率：$f_p=3400\text{Hz}$，通带最大衰减：$R_p=3\text{dB}$

阻带截止频率：$f_s=4000\text{Hz}$，阻带最小衰减：$A_s=40\text{dB}$

解：■ 建模

低通巴特沃斯模拟滤波器的系统函数的一般形式如下

$$H_a(s)=\Omega_c\Big/\prod_{k=0}^{N-1}(s-s_k)$$

$$\text{极点}\quad s_k=\Omega_c e^{j\pi\left(\frac{1}{2}+\frac{2k+1}{2N}\right)}$$

由此可见，低通巴特沃斯模拟滤波器的系统函数完全由阶数 N 和 3dB 截止频率 Ω_c 决定。而 N 和 Ω_c 是由滤波器设计指标决定的。其计算公式如下：

$$N=\frac{\lg k_{sp}}{\lg \lambda_{sp}},\quad \lambda_{sp}=\frac{\Omega_s}{\Omega_p},\quad k_{sp}=\sqrt{\frac{10^{R_p/10}-1}{10^{A_s/10}-1}}$$

$$\Omega_{c1}=\Omega_p(10^{0.1R_p}-1)^{-\frac{1}{2N}},\quad \Omega_{c2}=\Omega_s(10^{0.1A_s}-1)^{-\frac{1}{2N}}$$

取 $\Omega_c=\Omega_{c1}$，则所设计的滤波器的通带指标刚好满足，阻带指标富裕；

取 $\Omega_c=\Omega_{c2}$，则所设计的滤波器的阻带指标刚好满足，通带指标富裕。

MATLAB 工具箱函数 buttord，butter 就是根据以上关系式编写的。因此设计时就无需再记忆和使用这些公式了。

■ MATLAB程序q728.m

```
%低通巴特沃斯模拟滤波器设计
clear;close all
fp=3400;fs=4000;Rp=3;As=40;          % 输入滤波器指标
[N,fc]=buttord(fp,fs,Rp,As,'s')%计算阶数 N 和 3dB 截止频率 fc
[B,A]=butter(N,fc,'s'); %设计低通巴特沃斯模拟滤波器
[hf,f]=freqs(B,A,1024); %计算模拟滤波器频率响应,freqs 为工具箱函数
subplot(3,2,1);
plot(f,20*log10(abs(hf)/abs(hf(1))))
grid;xlabel('f / Hz');
ylabel('幅度(dB)')
axis([0,4000,-40,5])
line([0,4000],[-3,-3]);
line([3400,3400],[-90,5])
```

■ 程序运行结果

数字结果

```
N =29, fc =3.4127e+003
```

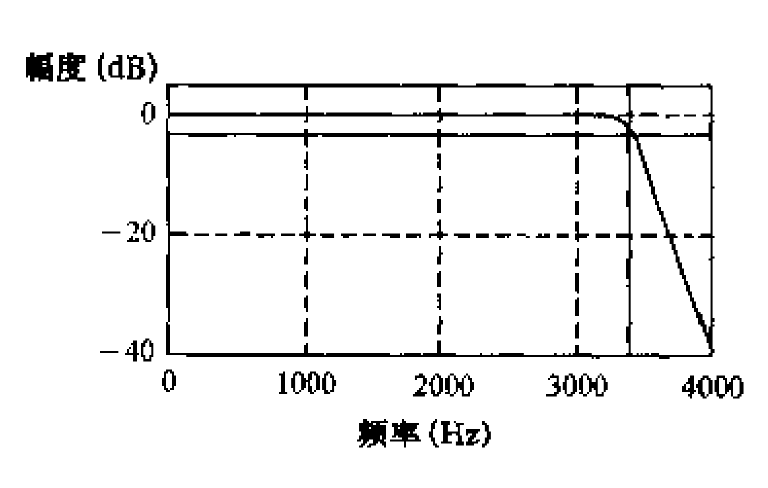


图 7.31　设计滤波器的频率特性

这个 29 阶的滤波器系数 A 和 B 各有 30 项。其频率特性如图 7.31 所示。图中幅频曲线在 3400Hz 处为-3dB；4000Hz 处的衰减为-40dB，说明刚好满足指标。

【例 7.29】 模拟低通转换为数字低通滤波器

已知一模拟滤波器的系统函数为

$$H_a(s)=\frac{1000}{s+1000}$$

分别用脉冲响应不变法和双线性变换法将 $H_a(s)$转换成数字滤波器系统函数 $H(z)$，并图示 $H_a(s)$和 $H(z)$的幅频响应曲线。分别取采样频率 F_s = 1000Hz 和 F_s = 500Hz，观察脉冲响应不变法中存在的频率混叠失真和双线性变换法中存在的非线性频率失真。

解：■ 建模

模拟滤波器离散化的基本方法有脉冲响应不变法和双线性变换法。

● 脉冲响应不变法及 impinvar 函数

对只有单阶极点的 N 阶模拟滤波器 $H_a(s)$，可用部分分式展开为

$$H_a(s)=\sum_{k=1}^{N}\frac{A_k}{(s-s_k)} \tag{7.18}$$

其中，s_k 为 $H_a(s)$的单阶极点。

则脉冲响应不变法取

$$H(z)=\sum_{k=1}^{N}\frac{A_k}{(1-e^{s_kT}z^{-1})} \tag{7.19}$$

为其近似离散化结果。对多阶极点情况可参考数字信号处理方面的书。MATLAB 工具箱函数 impinvar 可实现以上计算，格式为

➢ [Bz, Az] = impinvar(B, A, Fs)　　B 和 A 分别为（7.16）式表示的模拟滤波器的系数；Bz 和 Az 分别为（7.17）式表示的数字滤波器系数。Fs 为采样频率（Hz），缺省值为 Fs=1Hz。

● 双线性变换法函数 bilinear

双线性变换法的原理是用 $s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$代换 $H_a(s)$中的 s 值，得到 $H(z)$。bilinear 函数用来实现这个转换。其使用格式为

➢ [Bz, Az] = bilinear(B, A, Fs)

脉冲响应不变法的缺点是存在频率混叠失真。双线性变换法可完全消除频率混叠失真；缺点是存在非线性频率失真，所以适合设计片段常数频响特性的滤波器。

■ MATLAB程序q729.m

```
% 用脉冲响应不变法和双线性变换法将模拟滤波器离散化
clear; close all
```

```
b=1000; a=[1, 1000];
w=[0: 1000*2*pi];                      % 设定模拟频率
[hf, w]=freqs(b, a, w);            % 计算模拟滤波器频响函数
subplot(2, 3, 1)                   % 画出模拟滤波器幅频特性
plot(w/2/pi, abs(hf)); grid; title(' (a) |Ha(jf)|')
xlabel('f  (Hz)'); ylabel('幅度');
title(' (a) 模拟滤波器频响特性');
Fs0=[1000, 500];
for m=1: 2
   Fs=Fs0(m)        % T=0.001s 及 T=0.002s
   [d, c]=impinvar(b, a, Fs)    % 用 impinvar 函数实现离散化
   [f, e]=bilinear(b, a, Fs)    % 用 bilinear 函数实现离散化
   wd=[0: 512]*pi/512;          % 设定数字归一化频率
   hw1=freqz(d, c, wd);         % 计算数字滤波器频响函数
   hw2=freqz(f, e, wd);
   % 画出数字滤波器幅频特性
   subplot(2, 3, 2); plot(wd/pi, abs(hw1)/abs(hw1(1))); hold on
   subplot(2, 3, 3); plot(wd/pi, abs(hw2)/abs(hw2(1))); hold on;
end
```

■ **程序运行结果**

为了节省空间，把屏幕显示数据写在一行内。

Fs= 1000; d= 1; c=[1.0000 -0.3679]; f=[0.3333 0.3333]; e=[1 -0.3333]

Fs= 500; d=2; c=[1 0.1353]; f=[0.5 0.5]; e=[1 0]

图形结果如图 7.32 所示。由图 7.32(b)可见，对脉冲响应不变法，采样频率 Fs 越高(T 越小)，混叠越小；由图 7.32(c)可见，对双线性变换法，无频率混叠，但存在非线性失真。

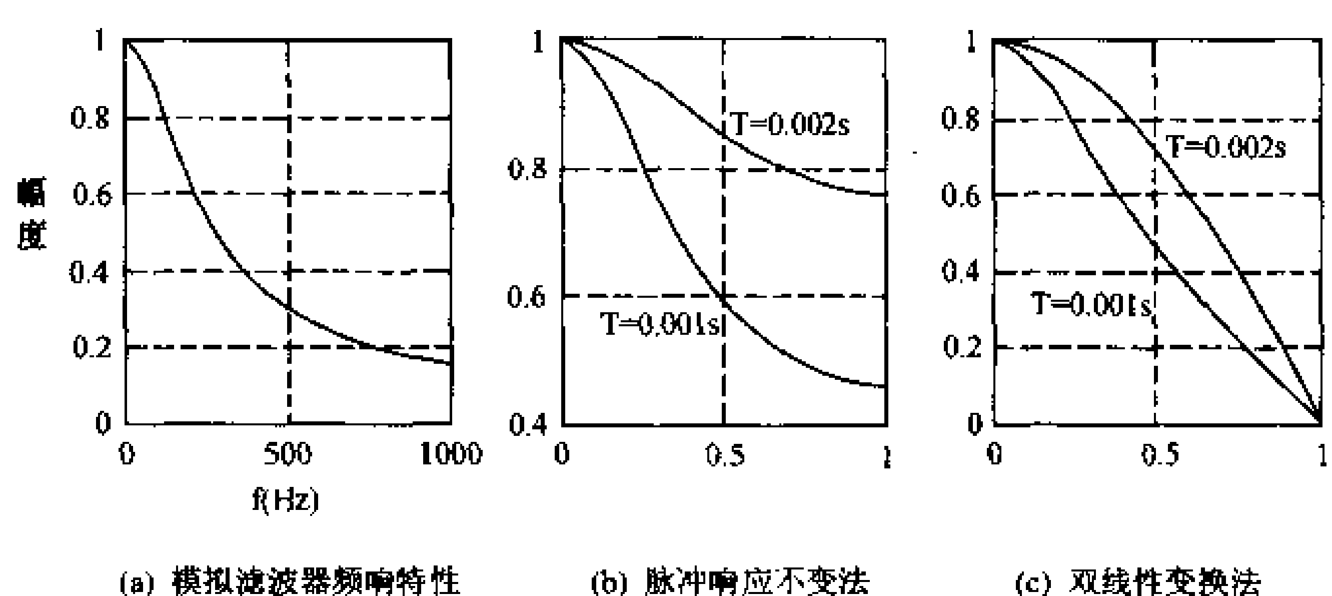


图 7.32 将连续系统离散化后频率特性的变化

【例 7.30】 切比雪夫 II 型低通数字滤波器设计

设计一个切比雪夫 II 型低通数字滤波器，指标如下。

通带边界频率：$\omega_p = 0.2\pi$，通带最大衰减：$R_p = 1$ dB

阻带截止频率：$\omega_s = 0.4\pi$，阻带最小衰减：$R_s = 80$ dB

解：切比雪夫 I 型滤波器通带内为等波纹，阻带内单调下降；切比雪夫 II 型滤波器通

带内为单调下降，阻带内等波纹。阶数相同时，切比雪夫 I 型过渡带比切比雪夫 II 型窄。调用 cheb2ord 函数和 cheby2 函数使切比雪夫 II 型设计变得非常简单。

先用[N, wc] = Cheb2ord(wp, ws, Rp, Rs)求出[N, wc]，提供函数 cheby2 的输入变元，再由[B, A] = cheby2(N, Rp, wc)设计切比雪夫 II 型数字滤波器。其返回值 B 和 A 分别为 $H(z)$的分子和分母多项式系数。

对切比雪夫 I 型滤波器，同样有相应的工具箱函数 cheb1ord 和 cheby1。

■ **MATLAB程序 q730.m**

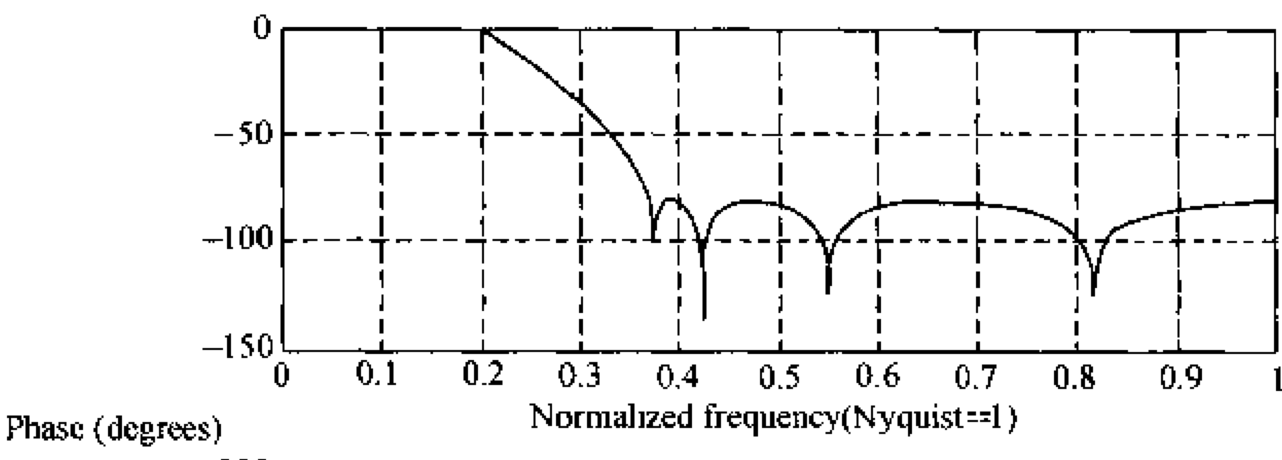
```
%切比雪夫II型低通数字滤波器设计
clear;close all
wp=0.2;ws=0.4;Rp=1;Rs=80;              % 输入指标
[N,wc]=cheb2ord(wp,ws,Rp,Rs)           % 求滤波器阶次
[B,A]=cheby2(N,Rs,wc)                  % 设计滤波器,得出系数
freqz(B,A)                             % 无左端变量时自动画频率特性图
```

■ **程序运行结果**

如图 7.33 所示。输出滤波器参数如下：

```
N =    8
wc =   0.3682
B =  Columns 1 through 7
    0.0011    0.0008    0.0026    0.0024    0.0032    0.0024    0.0026
  Columns 8 through 9
    0.0008    0.0011
A =  Columns 1 through 7
    1.0000   -4.3811    8.9294  -10.8479    8.5205   -4.4046    1.4577
    Columns 8 through 9
```

将系数向量 **B** 和 **A** 代入（7.17）式可写出 $H(z)$的表达式。



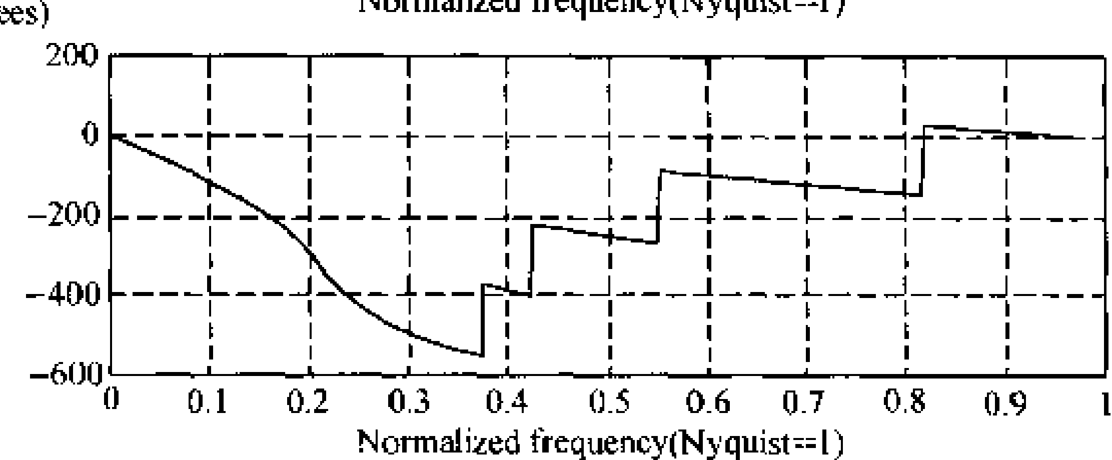


图 7.33　八阶切比雪夫 II 型低通滤波器幅频与相频曲线（R_s=80dB）

【例 7.31】 椭圆带通数字滤波器设计

设计一个椭圆带通数字滤波器，要求如下

$$\omega_{lp}=0.25\pi\text{，}\quad \omega_{ls}=0.15\pi$$

$$\omega_{up}=0.45\pi\text{，}\quad \omega_{us}=0.55\pi$$

$$R_p=1\ \text{dB},\quad R_s=60\ \text{dB}$$

解：调用 ellipord 函数和 ellip 函数，代入参数

wp = [0.25，0.45]；ws = [0.15，0.55]；

Rp = 1；Rs = 60;

即可得到本题的程序。

■ MATLAB程序q731.m

```
%直接设计带通数字椭圆滤波器
clear;close all
Wp=[0.25,0.45];Ws=[0.15,0.55];
Rp=0.1;Rs=60;
[N,wc]=ellipord(Wp,Ws,Rp,Rs)
[b,a]=ellip(N,Rp,Rs,wc)
[hw,w]=freqz(b,a);
subplot(3,2,1);
plot(w/pi,20*log10(abs(hw)));grid
axis([0,1,-80,5]);xlabel('ω/π');
ylabel('幅度（dB）')
subplot(3,2,3);
plot(w/pi,angle(hw));
grid;axis([0,1,-pi,pi])
xlabel('ω/π');ylabel('相位（rad）')
```

■ 程序运行结果

N，wc，b 和 a 的结果如下。

```
N =   6
wc =  0.2500    0.4500
b =  Columns 1 through 7
    0.0036   -0.0120    0.0213   -0.0300    0.0382   -0.0409    0.0403
  Columns 8 through 13
   -0.0409    0.0382   -0.0300    0.0213   -0.0120    0.0036
a =  Columns 1 through 7
    1.0000   -4.9576   14.7520  -30.3398   48.1073  -60.2322   61.3145
  Columns 8 through 13
  -50.5555   33.8742  -17.8964    7.2817   -2.0421    0.3460
```

2N 阶带通椭圆滤波器幅频及相频特性如图 7.34 所示。

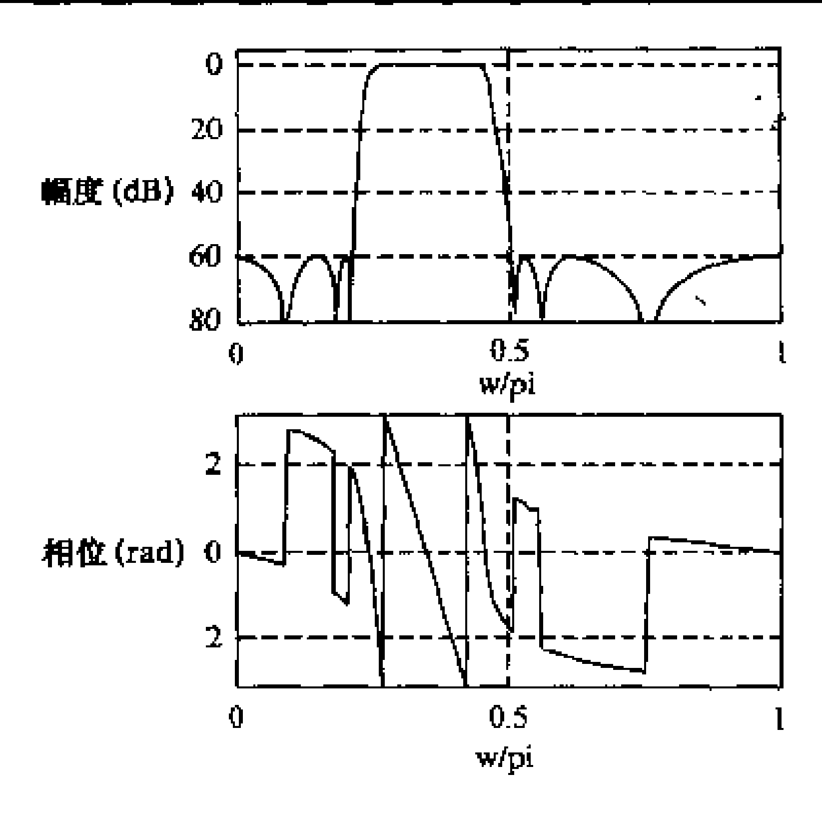


图 7.34　椭圆带通数字滤波器频率特性

比较 b，a 和 N 可知，带通滤波器是 2N 阶的。读者可修改程序，用 Butterworth 滤波器实现，并比较两种滤波器的阶数和频响特性曲线。

【例 7.32】 高通和带通巴特沃斯数字滤波器设计

已知四阶归一化低通巴特沃斯模拟滤波器系统函数为

$$H_a(s)=\frac{1}{s^4+2.6131\,s^3+3.4142\,s^2+2.6131\,s+1}$$

用双线性变换法从 $H_a(s)$设计 3dB 截止频率为 $\omega_c=\pi/2$ 的四阶高通巴特沃斯数字滤波器 $H_{hp}(z)$，并图示$|H_{hp}(e^{j\omega})|$ （设采样周期 $T=1s$）。

用双线性变换法从 $H_a(s)$设计四阶带通巴特沃斯数字滤波器 $H_{Bp}(z)$，并图示$|H_{Bp}(e^{j\omega})|$ （设采样周期 T=1s）。指标如下

$$\omega_{lc}=0.35\pi\ ,\qquad \omega_{uc}=0.65\pi$$

解：■ 建模

本题主要涉及三个问题：

（1）由数字滤波器指标求相应的模拟滤波器指标；

（2）模拟滤波器频率变换（因为已给定阶数和模拟滤波器归一化低通原型）；

（3）由相应的模拟滤波器到数字滤波器转换（双线性变换法）。

由例 7.29 可见，调用 bilinear 函数将模拟滤波器转换成数字滤波器非常容易。本题给定了数字滤波器指标，所以，首先要设计出与该指标相应的四阶 Butterworth 模拟滤波器。然后，调用 bilinear 函数将其转换成数字滤波器即可。应当特别注意，对双线性变换法，由数字边界频率求相应的模拟边界频率时，一定要考虑预畸变矫正。只有这样，最终设计结果才能满足所给指标。

问题（1）的解

● 设计高通数字滤波器时，相应的模拟高滤波器 3dB 截止频率为

$$\Omega_c=\frac{2}{T}\mathrm{tg}\left(\frac{\varpi_c}{2}\right)$$

● 设计带通数字滤波器时，相应的模拟带通滤波器 3dB 截止频率为

$$\Omega_{lc}=\frac{2}{T}\mathrm{tg}\left(\frac{\varpi_{lc}}{2}\right)\quad,\qquad \Omega_{uc}=\frac{2}{T}\mathrm{tg}\left(\frac{\varpi_{uc}}{2}\right)$$

问题（2）的解

可调用 MATLAB 频率变换函数 lp2lp，lp2hp，lp2bp，分别实现从模拟低通到模拟低通、高通、带通、带阻的频率变换。其中本题用到的 lp2hp 和 lp2bp 的格式及简要说明如下（其他通过 help 命令查看）。

➢ [Bt,At] = lp2hp(B, A, wc)　将系数向量为 B 和 A 的模拟滤波器归一化低通原型(3dB 截止频率为 1rad/s）变换成 3dB 截止频率为 wc 的高通模拟滤波器，返回高通模拟滤波器系数向量 Bt 和 At。

➢ [Bt, At] = lp2bp(B, A, wo, Bw)　将系数向量为 B 和 A 的模拟滤波器归一化低通原型变换成中心频率为 wo，带宽为 Bw 的带通模拟滤波器。返回带通模拟滤波器系数向量 Bt 和 At。

其中，$\omega_0=\sqrt{\Omega_{lc}\Omega_{uc}}$　，　$B_\omega=\Omega_{uc}-\Omega_{lc}$

由以上原理可编写如下程序。

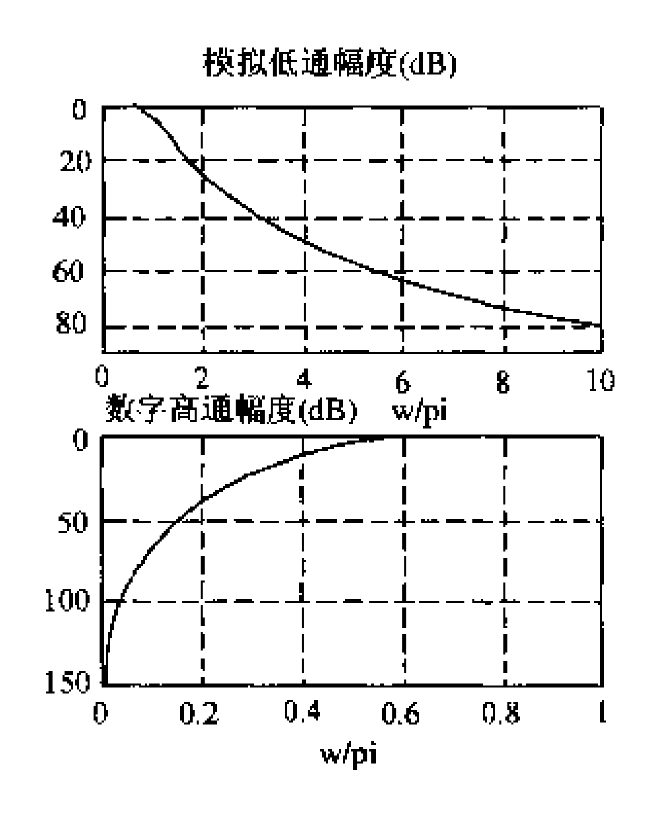
■ **MATLAB程序q732.m**

```
%用双线性变换法设计数字高通和带通滤波器
clear;close all
T=1;wch=pi/2;                          % T 为采样间隔,wch 为数字高通 3dB 截止频率
wlc=0.35*pi;wuc=0.65*pi;               % wlc,wuc:数字带通 3dB 截止频率
B=1;A=[1,2.6131,3.4142,2.6131,1];
[h,w]=freqs(B,A,512);                           % 求原归一化模拟滤波器的频率响应
subplot(3,2,1);plot(w,20*log10(abs(h)));        % 画模拟滤波器幅频特性
grid;axis([0,10,-90,0])
xlabel('ω/π');ylabel('模拟低通幅度(dB)')
%(1)设计高通
omegach=2*tan(wch/2)/T;               % 预畸变求模拟高通 3dB 截止频率
[Bhs,Ahs]=lp2hp(B,A,omegach);         % 模拟域低通转换为高通系数
[Bhz,Ahz]=bilinear(Bhs,Ahs,1/T);      % 模拟转换为数字高通系数向量
[h,w]=freqz(Bhz,Ahz,512);             % 求并画数字滤波器幅频特性
subplot(3,2,3);plot(w/pi,20*log10(abs(h)));
grid;axis([0,1,-150,0])
xlabel('ω/π');ylabel('数字高通幅度(dB)')
%(2)设计带通
omegalc=2*tan(wlc/2)/T;       % 预畸变求滤波器通带低端截止频率
omegauc=2*tan(wuc/2)/T;       % 预畸变求滤波器通带高端截止频率
wo=sqrt(omegalc*omegauc);Bw=omegauc-omegalc;
[Bbs,Abs]=lp2bp(B,A,wo,Bw);  %        % 模拟域低通转换为带通系数
[Bbz,Abz]=bilinear(Bbs,Abs,1/T);      % 模拟转换为数字带通系数向量
[h,w]=freqz(Bbz,Abz,512);             % 求并画数字滤波器幅频特性
subplot(3,2,4);plot(w/pi,20*log10(abs(h)));
```

```
grid;axis([0,1,-150,0])
xlabel('ω/π');ylabel('数字带通幅度(dB)')
```

■ **程序运行结果**

如图 7.35 所示。



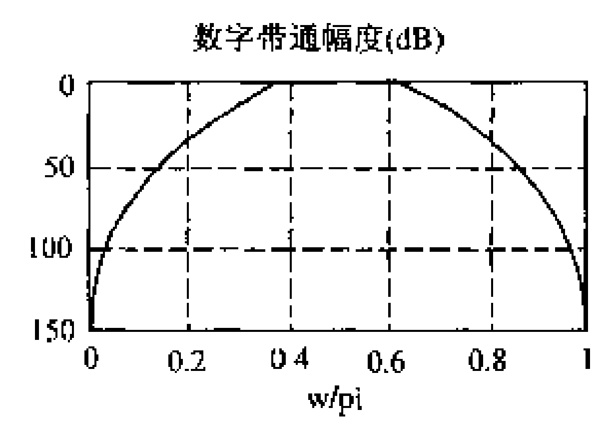


图 7.35　程序 q732.m 设计结果

第 8 章 MATLAB 在自动控制原理中的应用

自动控制系统通常是由基于机、电、光、化、热等不同物理原理的部件组成的复杂系统，用它来实现对某种物理量的精确控制。用统一的数学方法对系统进行描述时，可分为线性系统和非线性系统。因为非线性系统领域太宽，有无数不同的数学描述，没有也不可能有统一的通用解法，所以在课程中讨论得比较深入的是线性系统的理论。在实际应用中，凡是可能的场合下，人们往往也利用小偏差线性化的方法，把某些非线性系统近似为线性系统来求解。所以在 MATLAB 中，也着重于线性系统的算法。在线性系统中，又分别按连续系统和离散系统来解，不过，这两类系统的算法有许多相似点，往往可用同样的 MATLAB 函数来实现。

在线性系统中，我们又着重于线性时不变（Linear Time Invariant—LTI）系统，或称定常线性系统。这类系统可以用常系数线性微分方程来描述，而常系数线性微分方程是有严格的解析解的。无论是古典或现代控制理论分析方法，对于同一个系统，可以根据研究者的习惯和问题的性质，采取这两种分析方法之一对它进行描述和分析。其实这两种方法是统一的，具有公共的理论基础。它们可以互相变换。在没有良好的计算工具之前，这种变换靠手算来进行，非常麻烦，控制理论中有相当的篇幅就是用来介绍计算或做图的技巧的。借助于计算机和 MATLAB 等软件，可以快速地在各种描述和求解方法之间转换，节省大量的计算时间，避免计算错误，从而把注意力集中在概念的思考上。这对掌握好控制理论，将是非常有益的。

其实，在本书第 6 章中，已经详细讨论了一般线性系统的求解方法。线性控制系统是一般线性系统的子集，那些方法当然也适用于控制系统，它们是线性控制理论的基础。线性控制理论的特殊之处如下：

（1）系统比较复杂。通常它是由很多不同数学模型的部件组成，对于各个领域的线性部件，其数学描述方法也有所不同，由各个部件的数学模型求出整个系统的数学模型是一个研究重点。而在研究系统特性的同时，还要研究系统中各个部件的状态。因此要注重多变量系统的方法，如结构图、信号流图、状态空间法等。

（2）控制系统是由很多部件组成的闭环反馈系统。需要研究环节的特性如何影响系统的特性。换句话说，就是要着重研究开环特性如何影响系统的闭环特性。这是一般的线性系统理论中的非重点问题，但在本章要着重讨论。

（3）控制系统理论是一门技术科学，它更接近于工程。有许多工程中的特殊要求要满足。例如误差和精度、动态响应的速度、部件的功耗和饱和、内部噪声和外部干扰等等。许多控制系统又是价值很高的大系统，如飞行控制系统，生产过程控制系统等，微小的改进或优化可以带来很大的效益，所以控制系统理论得到了工业和国防部门的大量投资，促进了它的迅速发展和深入。近三十年来它已成为线性系统理论中发展最快的一个门类。在 MATLAB 中，用于特定控制理论的工具箱有将近十个。这足以证明它的地位。

在本书的第 7 章以前，基本不用 MATLAB 中的工具箱，这样做有两个目的：①避免读者对原理的忽视；②提高读者的 MATLAB 编程能力。因此在进入本章之前，请读者务必先读懂第 6 章了解有关线性系统的表述方法、相互变换及求解的问题。

随着研究问题的日益复杂，再拒绝工具箱就不合适了。特别是控制系统（Control）工具箱，它一开始就为线性模型提供了一种特别简洁的数据结构，而所有的控制系统分析设计的工具箱命令都以这种新的面向对象的数据结构为基础。因此，本章将重点介绍控制系统工具箱。但读者仍应把理解原理放在主要的地位，不提倡“傻瓜”式的应用。

8.1　控制工具箱中的 LTI 对象

由 6.4 节可以看到，一个线性系统可以采取四种不同的方法进行描述，每种方法又需要几个参数矩阵，因此对系统进行调用和计算都很不方便。根据软件工程中面向对象的思想，MATLAB 通过建立专用的数据结构类型，把线性时不变系统的各种模型封装成为统一的 LTI（Linear Time Invariant）对象，它在一个名称之下包含了该系统的全部属性，大大方便了系统的描述和运算。本节着重介绍这种描述线性系统的方法。版本 3.x 以前的控制工具箱，采用的是老方法，从版本 4.x 以后，其函数库和用法都作了很大的修改。所以这一节是本章的基础，即使会用原有控制工具箱的读者，也必须阅读这一节，才能掌握本章。

8.1.1　LTI 对象的类型和属性

MATLAB 控制系统工具箱中规定的 LTI 对象，包含了以下三种子对象：ss 对象、tf 对象和 zpk 对象，他们分别与状态空间模型、传递函数模型和零极增益模型相对应。每个对象都具有其属性和方法，通过对象方法可以存取或者设置对象的属性值，在控制系统工具箱中，这三种对象除了具有 LTI 的共同的属性（即子对象可以继承父对象的属性）外，还具有一些各自特有的属性。这些共同属性见表 8.1。

表 8.1　LTI 对象共有属性列表

属 性 名 称	意　　义	属性值的变量类型
Ts	采样周期	标量
Td	输入时延	数组
InputName	输入变量名	字符串单元矩阵（数组）
OutputName	输出变量名	字符串单元矩阵（数组）
Notes	说明	文本
Userdata	用户数据	任意数据类型

● 当系统为离散系统时，采样周期 Ts 给出了系统的采样周期，Ts = 0 或缺省时表示系统为连续时间系统，Ts = -1 表示系统是离散系统，但它的采样周期未定。

● 输入时延 Td 仅对连续时间系统有效，其值为由每个输入通道的输入时延组成的时延数组，缺省表示无输入时延。

● 输入变量名 InputName 和输出变量名 OutputName 允许用户定义系统输入输出的名称，其值为一字符串单元数组，分别与输入输出有相同的维数，可缺省。

● 说明 Notes 和用户数据 Userdata 用以存储模型的其他信息，常用于给出描述模型的文本信息，也可以包含用户需要的任意其他数据，可缺省。

三种对象的特有属性见表 8.2。

表 8.2　三种子对象特有属性列表

对 象 名 称	属 性 名 称	意　义	属性值的变量类型
tf 对象 (传递函数)	den	传递函数分母系数	由行数组组成的单元阵列
	num	传递函数分子系数	由行数组组成的单元阵列
	variable	传递函数变量	s，p，z，q，z^-1 中之一
zpk 对象 (零极增益)	k	增益	二维矩阵
	p	极点	由行数组组成的单元阵列
	variable	零极增益模型变量	s，p，z，q，z^-1 中之一
	z	零点	由行数组组成的单元阵列
ss 对象 (状态空间)	a	系数矩阵	二维矩阵
	b	系数矩阵	二维矩阵
	c	系数矩阵	二维矩阵
	d	系数矩阵	二维矩阵
	e	系数矩阵	二维矩阵
	StateName	状态变量名	字符串单元向量

每一类对象只含有自己的属性，这些属性中绝大部分前面已叙述过。num 是 6.4 节中的 f，den 是 6.4 节中的 g，只有 Variable 同属于前两类对象，它是用来显示系统函数中频率变量的。缺省时连续系统为 s，离散系统为 z，对 DSP（数字信号处理）式传递函数为 z^-1，p 和 q 留作用户自行规定。

ss 对象的属性 e 用于“描述状态空间模型”中左端（导数端）的系数。在标准状态空间模型中，它是单位矩阵 eye(n)。ss 对象的属性 StateName 用于定义状态空间模型中每个状态的名称。

8.1.2 LTI 模型的建立

各种 LTI 对象模型都可以通过一个相应函数来建立，这种函数有五个，见表 8.3。

表 8.3　生成 LTI 模型的函数

函数名称及基本格式	功　能
dss(a，b，c，d，…)	生成（或将其他模型转换为）描述状态空间模型
filt(num，den，…)	生成（或将其他模型转换为）DSP 形式的离散传递函数
ss(a，b，c，d，…)	生成（或将其他模型转换为）状态空间模型
tf(num，den，…)	生成（或将其他模型转换为）传递函数模型
zpk(z，p，k，…)	生成（或将其他模型转换为）零极增益模型

其中 dss 和 ss 函数都生成状态空间模型（它包含了描述状态空间模型）；filt 函数生成的仍然是传递函数模型，它的存储变量仍是 num，den，不过自动取 z^{-1} 为显示变量，所以五种函数实际上生成的仍然是前面所说的三种对象模型。

表 8.3 中所列的基本格式给出了最低限度应输入的基本变元，这些变元后面还可以增加对象的属性参数。例如键入

```
s1=tf([3, 4, 5], [1, 3, 5, 7, 9])
```

得出

```
Transfer function:
      3 s^2 + 4 s + 5
---------------------------------
s^4 + 3 s^3 + 5 s^2 + 7 s + 9
```

如果键入

```
s2=tf([3, 4, 5], [1, 3, 5, 7, 9], 0.1, 'InputName', '电流', 'OutputName', '转速')
```

则根据规定，紧接着基本变元的第一个不加属性名称的变元表示采样周期，有了这个变元，就是离散系统。所以就自动以 z 作为传递函数变量来显示，得出

```
Transfer function from input "电流" to output "转速":
      3 z^2 + 4 z + 5
-------------------------------------
z^4 + 3 z^3 + 5 z^2 + 7 z + 9
Sampling time: 0.1
```

如果想要加入 0.1s 的时延变量，就要把时延属性名称 td 键入，这时候对象仍然是连续系统模型。键入

```
s3=tf([4, 5], [1, 5, 7, 9], 'td', 0.1,  'InputName', 'u', 'OutputName', 'y')
```

得出

```
Transfer function from input "u" to output "y":
       4 s + 5
---------------------------
s^3 + 5 s^2 + 7 s + 9
Input delay: 0.1
```

再来看一下用 filt 函数生成模型。键入

```
s4=filt([3, 4, 5], [1, 3, 5, 7, 9], 0.1)
```

得出

```
Transfer function:
         3 + 4 z^-1 + 5 z^-2
-------------------------------------------
1 + 3 z^-1 + 5 z^-2 + 7 z^-3 + 9 z^-4
Sampling time: 0.1
```

把系统 s2 与 s4 相比，这两个传递函数是不同的，它们之间差了一个因子 z^2。差别原因在于 filt 函数把分子分母系数向量里的第一项对齐（都是 z^{-1} 的零次项），分子比分母系数向量短的部分在后面补零。而 tf 函数把分子分母系数向量里的末项对齐（都是 z 的零次项），分子比分母系数向量短的部分在前面补零。这个差别因子的值取决于分母系数向量和分子系数向量长度之差。

这几个函数不但可以用来生成模型，而且可以进行模型的转化，比如键入

```
s5=ss(s1)
a =              x1          x2          x3           x4
          x1   -3.00000    -1.25000    -0.87500     -0.56250
```

```
          x2      4.00000     0           0           0
          x3      0           2.00000     0           0
          x4      0           0           2.00000     0
b =                 u1
          x1        1.00000
          x2        0
          x3        0
          x4        0
c =                 x1          x2          x3          x4
          y1        0           0.75000     0.50000     0.31250
d =                 u1
          y1        0
Continuous-time system.
```

这一语句把原来的传递函数模型 s1 转换为等价的状态空间模型 s5，如果不需显示转换后的模型参数，可以用分号结束语句。

可见，在建立对象模型之后，人们只要用 s1，s2，…中的一个变量就可以称呼一个系统，其中已包含了研究和计算该系统的全部数据。

再来看一个双输入单输出系统。

```
z = { [],  -0.5 };                   % 外括号是花括号
p = { 0.3,  [0.1+2i, 0.1-2i] };      % 外括号是花括号
k = [ 2,  3 ];                        % 外括号是方括号，说明是两个数字阵列
s6=zpk(z, p, k, -1)
```

前两行外括号是花括号，说明是单元阵列，两单元可以不同长，表示有不同数量的零极点。末行最后一个变元-1 表示定义的是采样系统，但采样周期未定。如果省略它，MATLAB 就误认为是连续系统了。有关单元阵列的概念可参阅本书 4.8 节。

系统对上述程序的反应为

```
Zero/pole/gain from input 1 to output:
  2
-------
(z-0.3)
Zero/pole/gain from input 2 to output:
    3 (z+0.5)
-------------------
(z^2 - 0.2z + 4.01)
Sampling time: unspecified
```

在上述程序中，若把各行中分割两单元的逗号改为分号，原来的双输入单输出就变成单输入双输出系统了。另外，再把末行中的-1 去掉，改成 s7 = zpk(z, p, k)，即变成连续系统。MATLAB 对修改后程序的反应为

```
Zero/pole/gain from input to output...
```

```
           2
#1:    -------
       (s-0.3)
          3 (s + 0.5)
#2:    ------------------
       (s^2 - 0.2s + 4.01)
```

得出的是输入到输出“#1:”和输出“#2:”的零极增益表示式，可见，它确实是一个连续系统。对任意多输入多输出(MIMO)系统，MATLAB 的规定是：不同行代表不同输出，不同列代表不同输入。其系统函数表现为一个输出数 Ny 乘以输入数 Nu 的系统函数矩阵。

表 8.4　对象属性的获取和修改函数

函数名称及基本格式	功　能
get(sys，'PropertyName'，数值，…)	获得 LTI 对象的属性
set(sys，'PropertyName'，数值，…)	设置和修改 LTI 对象的属性
ssdata，dssdata(sys)	获得变换后的状态空间模型参数
tfdata(sys)	获得变换后的传递函数模型参数
zpkdata(sys)	获得变换后的零极增益模型参数

8.1.3　对象属性的获取和修改

1．对象属性提取和修改的方法

■ 用 get 和 set 命令

这种方法可以看到模型中存储的全部属性并可对它们进行修改。例如键入

```
get(s1)
```

得到

```
num = {[0 0 3 4 5]}
den = {[1 3 5 7 9]}
Variable = 's'
Ts = 0
Td = 0
InputName = {''}
OutputName = {''}
Notes = {}
UserData = []
```

如果键入

```
get(s5)
```

则得到

```
a = [4 × 4 double]
b = [4 × 1 double]
c = [0 0.75 0.5 0.312]
d = 0
e = []
StateName = {4x1 cell}
```

```
Ts = 0
…（后面 Td, InputName 等与前同）
```

从这里可以看出，系统 s1 和 s5 虽然是等价的，但因为分别属于 tf 和 ss 模型，它们内部保存的属性参数是不同的。其差别主要反映在前几项上。因为模型不同，将来对它的运算方法和属性调用也不同，因此，这些模型类型必须加以区别。

在状态空间模型中，它的系数矩阵 a，b 并没有完全显示出来。要得到它，可以键入

```
s5.a
ans =  -3.0000      -1.2500      -0.8750      -0.5625
        4.0000       0            0            0
        0            2.0000       0            0
        0            0            2.0000       0
```

要修改这些系统的属性，可以用 set 命令。例如键入

```
set(s1, 'num', [0, 1, 2, 3, 4], 'den', [2, 4, 6, 8, 10])
```

再键入

```
get(s1)
```

得到

```
num = {[0 1 2 3 4]}
den = {[2 4 6 8 10]}
Variable = 's'
……
```

■ 用单元阵列的访问方法提取单项属性和对它单独赋值（参阅 4.8 节）

键入　`s1.num`

得到　`ans =    [1×5 double]`

并未显示具体值，再键入花括号下标 s1.num{:}，表示要访问单元阵列的全部内容。

得到　`ans =    0    1    2    3    4`

要修改这个属性，可键入

```
s1.num={[0, 5, 4, 3, 2]},
```

注意外括号是花括号，那是单元阵列的规定。

MATLAB 会把修改后的系统传递函数显示出来，得到

```
Transfer function:
   5 s^3 + 4 s^2 + 3 s + 2
-------------------------------------------
2 s^4 + 4 s^3 + 6 s^2 + 8 s + 10
```

再来看看零极增益模型 s6 的情况。它的主要属性是 z，p，k。键入

```
s6.p
ans =    [0.3000]    [2×1 double]
```

MATLAB 未给出 p 中第二个单元的具体内容，可再键入（注意外括号是花括号）

```
s6.p{2}
ans =    0.1000 + 2.0000i
         0.1000 - 2.0000i
```

也可以重新给它赋值（注意上面的答案中已用方括号表明它是一个 2×1 的数字阵列），键入

```
s6.p{2}=[0.5; 0.7]
```

MATLAB 会把修改后的系统函数（零极增益形式）显示出来，得到

```
Zero/pole/gain from input 1 to output:
   2
-------
(z-0.3)
Zero/pole/gain from input 2 to output:
   3 (z+0.5)
---------------
(z-0.5) (z-0.7)
Sampling time: unspecified
```

加上 8.1.1 节中介绍的用 tf，zpk，ss 等函数重新生成系统，所以共有了三种方法来设置对象属性。

2. 模型类型的参数转换和提取

在第 6 章 6.4 节中介绍过线性模型在状态空间、传递函数和零极增益三种表述方式之间的相互转换问题。当时采用的是以下的一些转换命令：ss2tf，sstzp，tf2zp，tf2ss，zp2tf，zp2ss 等。用这些命令时，输入变元中要键入相应的系数矩阵，不太方便。在采用 LTI 模型以后，就不再用这些命令来进行模型变换了，而用能直接调用系统的 LTI 名称的命令来实现这些转换。这些命令就是 dssdata，ssdata，tfdata 和 zpkdata，它们分别用来获得转换后的系统状态空间、传递函数和零极增益参数。与 ss，tf，zpk 命令的不同在于这些带 data 的命令仅仅用来转换参数，但并不生成新的系统。要显示和存储这些转换后的参数，左端必须列出相应数目的输出变元。例如对原有的系统 s1 和 s2，要求出 s1 的传递函数系数，可键入

```
[f1, g1]=tfdata(s1)
```

得到

```
f 1=    [1×5 double]
g 1=    [1×5 double]
```

再键入

```
f1{1}, g1{1} （注意用的是花括号）
ans =    0    5    4    3    2
ans =    2    4    6    8    10
```

要求出 s1 的零极增益系数，可键入

```
[z1, p1, k1, T1s]=zpkdata(s1)
```

得到

```
z1 =    [3×1 double]
p1 =    [4×1 double]
k1 =     2.5000
T1s =    0
```

再键入

```
z1{1}, p1{1}
ans =    -0.7293
```

```
    -0.0353 + 0.7397i
    -0.0353 - 0.7397i
ans =   0.2878 + 1.4161i
        0.2878 - 1.4161i
       -1.2878 + 0.8579i
       -1.2878 - 0.8579I
```

要求出 s2 的状态空间系数阵，可键入

```
[ a2, b2, c2, d2, Ts2, Td2 ] = ssdata (s2)
```

得到

```
a2 =    -3.0000   -1.2500   -0.8750   -0.5625
         4.0000    0         0          0
         0         2.0000    0          0
         0         0         2.0000     0
b2 =    1
        0
        0
        0
c2 =    0    0.7500   0.5000   0.3125
d2 =    0
Ts2 =   0.1000
Td2 =   [ ]
```

3. 模型类型的检验

➢ cs1=class(s1)　　得出系统的对象类型：cs1=tf，ss 或 zpk。

➢ isa(s1，tf)　　得出一个逻辑量，当 s1 属 tf 类型时为真，它等于 1，否则等于 0。

另外，还有用来检验模型是否连续(isct)、是否离散(isdt)、是否 SISO 系统(issiso)等的命令，可查看控制工具箱函数库表中“模型尺度和特征”部分。

8.1.4 LTI 模型的简单组合和运算符扩展

先讨论由两个环节组成的合成系统，两个环节可以有串联、并联和反馈三种情况。在例 6.19 中讨论过，现在来看用 lti 模型处理有什么不同。先假定两环节均为单输入单输出的系统 SA 和 SB。在控制系统工具箱里，合成系统的特性可以用下列语句实现。

- 两个环节串联 S=series(SA，SB)　　　　或　S=SA*SB
- 两个环节并联 S=parallel(SA，SB)　　　或　S=SA+SB
- A 环节前向，B 环节反馈 S=feedback(SA，SB)

这几个函数已经在 6.4 节中介绍过，但在这里使用时，只要输入环节的名称，不必输入其参数矩阵。下面讨论怎么把它们移模到几种不同的 LTI 对象中。以 feedback 函数为例，其实控制系统工具箱共有四个 feedback 函数，第一个曾在 6.4 节中介绍过，且在控制系统工具箱目录中能够查到，其他三个分别放在三种对象的方法库中。这些方法库的目录名以@开始，如@tf，@ss，@zpk，@lti（@lti 是三种 lti 对象公用部分，即父对象的库）等。MATLAB

通常是不去查询的，只有当其调用的数据属于这个特定类型的时候，才到相应的方法目录中去查询并优先调用。于是，执行这个命令的过程可以分为三步：

（1）判断输入对象的类型（用 class 命令），并提取它的参数（用??data 命令）；

（2）到这个对象的方法库中寻找相应的 feedback 函数；

（3）根据求出的组合后的参数生成新系统。

从控制原理的角度考虑，主要需弄清对几种不同对象模型，怎样编制这几个函数程序。传递函数法已在前面介绍过，这里只介绍零极增益和状态空间法，可参照图 6.18。

● **零极增益法**

串联：将 $H_A(s)$和 $H_B(s)$的零极增益式代入 $H(s)=H_A(s)H_B(s)$中，可以得知，合成系统的零极点为 A，B 两系统零极点的并集，即

z = [zA, zB]；p = [pA, pB]；k = kA*kB;

并联：将 $H_A(s)$和 $H_B(s)$的零极增益式代入 $H(s)=H_A(s)+H_B(s)$中，可以得知，合成系统的极点为 A，B 两系统极点的并集，即 p=[pA，pB]；但其零点没有简单的表示式，只能按传递函数法中求出的 f 求根。

反馈：从反馈公式

$$H(s)=\frac{f_A(s)g_B(s)}{f_A(s)f_B(s)+g_A(s)g_B(s)}$$

可以观察到，合成系统的零点为系统 A 的零点加系统 B 的极点，即

z = [zA, pB]

而合成系统的极点则要经过多项式相乘、相加并求根等多个运算步骤才能得到，即

p = roots(polyadd(conv(fA, fB)，conv(gA, gB)))

● **状态空间法**

对系统 A，有状态方程

$$\dot{\boldsymbol{X}}_A=\boldsymbol{A}_A\boldsymbol{X}_A+\boldsymbol{B}_A\boldsymbol{U}_A$$
$$\boldsymbol{Y}_A=\boldsymbol{C}_A\boldsymbol{X}_A+\boldsymbol{D}_A\boldsymbol{U}_A$$

对系统 B，有状态方程

$$\dot{\boldsymbol{X}}_B=\boldsymbol{A}_B\boldsymbol{X}_B+\boldsymbol{B}_B\boldsymbol{U}_B$$
$$\boldsymbol{Y}_B=\boldsymbol{C}_B\boldsymbol{X}_B+\boldsymbol{D}_B\boldsymbol{U}_B$$

串联：$\boldsymbol{U}=\boldsymbol{U}_A$，$\boldsymbol{Y}=\boldsymbol{Y}_B$，$\boldsymbol{Y}_A=\boldsymbol{U}_B$，在这些方程中，消去 $\boldsymbol{Y}_A$ 及 $\boldsymbol{U}_B$，合成系统的状态方程可表为

$$\dot{\boldsymbol{X}}=\begin{bmatrix}\dot{\boldsymbol{X}}_A\\ \dot{\boldsymbol{X}}_B\end{bmatrix}=\boldsymbol{AX}+\boldsymbol{BU}=\begin{bmatrix}\boldsymbol{A}_A & 0\\ 0 & \boldsymbol{A}_B\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_A\\ \boldsymbol{X}_B\end{bmatrix}+\begin{bmatrix}\boldsymbol{B}_A & 0\\ 0 & \boldsymbol{B}_B\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_A\\ \boldsymbol{Y}_A\end{bmatrix}$$
$$\boldsymbol{Y}=\boldsymbol{C}_B\boldsymbol{X}_B+\boldsymbol{D}_B\boldsymbol{Y}_A$$

最后得到合成系统的状态方程可表为

$$\dot{\boldsymbol{X}}=\boldsymbol{AX}+\boldsymbol{BU}$$
$$\boldsymbol{Y}=\boldsymbol{CX}+\boldsymbol{DU}$$

其中

$$A=\begin{bmatrix} A_A & 0 \\ B_B C_A & A_B \end{bmatrix},\quad B=\begin{bmatrix} B_A \\ B_B D_A \end{bmatrix}$$

$$C=[D_B C_A, C_B],\quad D=D_A D_B$$

并联：系统 A，B 的状态方程仍同上。只是在并联系统中，$U=U_A=U_B$；$Y=Y_A+Y_B$，在这些方程中，消去 Y_A 及 Y_B，合成系统的状态方程可表为

$$\dot{X}=AX+BU$$

$$Y=CX+DU$$

其中

$$A=\begin{bmatrix} A_A & 0 \\ 0 & A_B \end{bmatrix},\quad B=\begin{bmatrix} B_A \\ B_B \end{bmatrix}$$

$$C=[C_A, C_B],\quad D=D_A+D_B$$

反馈：反馈系统状态方程的联结关系为

$$Y=Y_A=U_B;\quad U=Y_B+U_A$$

在 $D_A=D_B=0$ 的物理系统中，合成后系统的状态方程系数阵如下。

$$A=\begin{bmatrix} A_A & B_A C_B \\ B_B C_A & A_B \end{bmatrix}$$

$$B=\begin{bmatrix} B_A \\ 0 \end{bmatrix}$$

$$C=[C_A\ \ 0]$$

$$D=0$$

旧版本控制工具箱中关于反馈变换还有一个名为 cloop 的函数，它是 feedback 函数当系统 B 为单位直通特性时的特例。新版本控制工具箱中为了减少函数的数目，已把 cloop 列入取消的目录。

这里介绍的是基本的编程原理，实际上作为一个商品化的软件产品，程序的编写要考虑到各种各样的复杂情况，比如对输入数据的检验、输入有错误时如何向用户提示、多输入多输出系统的联接等等，所以实际的程序要复杂得多。即使以调用的方法来说，前面介绍的几种调用形式也是最基本的，在多输入多输出系统中，调用上述函数还必须增加输入变量和输出变量的编号，例如

➢ 串联： S = series(SA，SB，outputA，inputB)

后两个变元为互相串接的两系统输出编号和 B 系统输入编号。

➢ 并联： S = parallel(SA，SB，InputA，InputB，OutputA，OutputB)

前两个变元为互相并接的两系统输入编号，后两个变元为互相并接的两系统输出编号。

➢ 反馈： S = feedback(SA，SB，feedout，feedin，sign)

SA，SB 后的两个变元为 A 系统输出反馈编号和 B 系统输入编号，末变元表示正负反馈，负反馈可缺省。

前面提到，两个系统的串联可表述为两个 LTI 对象的相乘 s=s1*s2，两个系统的并联可表述为两个 LTI 对象的相加 s=s1＋s2，扩展的运算符（Overheaded operators）不仅使变换的表达更加简洁，而且可以把矩阵运算的法则也用到对象和系统函数中来。例 6.20 中谈到的信号流图的算法就因此可以推广到 LTI 对象。这也是面向对象编程方法优越性的一个很好的

例了。

实现运算符扩展的主要方法是在对象的方法库中增加一个规定的运算符函数。比如加法符“+”对应的函数名规定为 plus.m，乘法符“*”对应的函数名规定为 mtimes.m。用这些程序确定了对于这类对象加法或乘法所实现的运算，也就是并联或串联时要作的运算。三种 LTI 对象的相加相乘算法各异，因此，在三个方法库中都有各自的 plus.m 和 mtimes.m 函数文件。运算符远不止这两种，在控制工具箱函数库中，还列出了减法（-）、左除（\）、右除（/）、求逆 inv、转置（.'）、共轭转置（'）、幂次（^）等多种扩展的运算符。可以用运算符写出反馈连接的算式

S = feedback(SA, SB)

等价于　　S = (1+SA*SB)\SA

其中用到了加法符“+”、乘法符“*”和左除（\）（也就是求逆 inv）。

这些 LTI 对象的运算符是以多项式计算为基础的，因此，难以应用到带时延 Td 的系统。例如键入

```
s1=tf(2, [1, 1], 'Td', 0.2)        % 建立一个带时延 Td=0.2 的简单系统
s2=feedback(s1, 1)                 % 加一个单位负反馈
```

得出

```
??? Error using ==> tf/feedback
FEEDBACK cannot handle time delays.
```

因此时延环节必须要用一个 N 次多项式来近似，MATLAB 才能处理，称为 Pade 近似。该多项式的分子分母系数向量可用语句[numd，dend] = pade(Td，N)求得，键入

```
[numd, dend] = pade(0.2, 3)
```

得

```
numd =   -1    60    1500    15000
dend =    1    60    1500    15000
```

通常并不需要求出系数，直接把含有时延的环节 s1 变换一下即可。设近似后的环节为 spd1，用的是三次多项式，Td 已包含在 s1 的属性中，无需再输入。因此可键入

```
spd1=pade(s1, 3)
```

得出

```
Transfer function:
     -s^3 + 60 s^2 - 1500 s + 1.5e004
--------------------------------------------------
s^4 + 61 s^3 + 1560 s^2 + 1.65e004 s + 1.5e004
```

再键入

```
s2=feedback(s1, 1)
```

就得出闭环传递函数

```
Transfer function:
     -s^3 + 60 s^2 - 1500 s + 1.5e004
--------------------------------------------------
s^4 + 60 s^3 + 1620 s^2 + 1.5e004 s + 3e004
```

这样带时延系统的其他特性也都可以分析了。

8.1.5 复杂模型的组合

1．信号流图

遇到由大量环节交叉联接的系统，计算方法之一是画成信号流图，用梅森公式来求解。用 MATLAB 来辅助时，不便用梅森公式，6.4 节给出了规范的易于编程的方法和简明的公式。这里再简要地重复一下。

设信号流图中有 K_i 个输入节点，K 个中间和输出节点，它们分别代表输入信号 u_i（i=1，2，…，K_i）和系统状态 x_j（j=1，2，…，K）。信号流图代表它们之间的联接关系。用系统函数表示后，任意状态 x_j 可以表为 u_i 和 x_j 的线性组合

$$x_j = \sum_{K=1}^{K} r_{jK} x_K + \sum_{i=1}^{Ki} p_{ji} u_i$$

用矩阵表示，可写成

$$X = RX + PU$$

其中，$X = [\,x_1; x_2; \ldots; x_K\,]$ 为 K 维状态列向量，$U=[u_1; u_2; \ldots; u_{Ki}]$ 为 K_i 维输入列向量，R 为 $K\times K_i$ 阶的传输矩阵，P 为 $K\times K_i$ 阶的输入矩阵，R 和 P 的元素 r_{ji} 和 p_{jK} 是各环节的系统函数。

由此可得

$$(I - R)X = PU$$

$$X = (I - R)^{-1} PU$$

因此，系统的传递函数矩阵为 $H = (I-R)^{-1} P$，这个简明的公式就等价于梅森公式。只要写出 P 和 R，任何复杂系统的传递函数都可用这个简单的式子求出。

在 6.4 节中曾经指出，用这个式子存在的困难是，公式中用到的是普通的矩阵乘法和加法，如何将它推广到传递函数或其他系统函数。当时利用 MATLAB 的符号运算 (Symbolic) 工具箱解决了这个问题。现在，利用 LTI 对象和它的扩展运算符，这个难题也得到了解决。由于可以直接调用环节的 LTI 名称作为传输矩阵 R 的一个单元，可对它作矩阵的乘法、加法和求逆，因而可以利用与例 6.20 同样的程序来解决问题。例 8.4 给出了比较。

MATLAB 控制工具箱中没有给出解信号流图的函数，这个公式是作者推导的。它特别适合于由单输入单输出环节组成的系统，用它可以得到传递函数和零极增益的表示式。而用 MATLAB 控制工具箱中复杂系统的简化方法只能得出状态空间的模型，下面将讲述这个问题。

2．复杂系统状态方程的合成

任意复杂的线性环节组成的系统，可以推导出它的普遍的状态方程表示式。

设系统有 L 个环节，其状态方程为

$$\begin{aligned} \dot{x}_i &= A_i x_i + B_i u_i \\ y &= C_i x_i + D_i u_i \end{aligned} \qquad i=1, 2, \cdots, L \tag{8.1}$$

其中 x 为 n_i 维状态向量，n_i 为第 i 个环节的阶数。现设整个合成系统的状态向量为

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_L \end{bmatrix} \tag{8.2}$$

显然 x 的维数 $n=\sum_{i=1}^{L}n_i$，先不考虑各环节的相互联接，只把各个环节并列出来，组成一个大的互不相关的系统方程

$$\begin{aligned}\dot{x}&=\overline{A}x+\overline{B}u\\ y&=\overline{C}x+\overline{D}u\end{aligned}\tag{8.3}$$

其中

$$\overline{A}=\begin{bmatrix}A_1 & & 0\\ & \ddots & \\ 0 & & A_L\end{bmatrix},\ \overline{B}=\begin{bmatrix}B_1 & & 0\\ & \ddots & \\ 0 & & B_L\end{bmatrix},\ \overline{C}=\begin{bmatrix}C_1 & & 0\\ & \ddots & \\ 0 & & C_L\end{bmatrix},\ \overline{D}=\begin{bmatrix}D_1 & & 0\\ & \ddots & \\ 0 & & D_L\end{bmatrix}\tag{8.4}$$

如果各环节都是单输入单输出的，则各矩阵的大小 $\overline{A}$ 为 $n\times n$ 阶，$\overline{B}$ 为 $n\times L$ 阶，$\overline{C}$ 为 $L\times n$ 阶，$\overline{D}$ 为 $L\times L$ 阶。输入 u 和输出 y 的长度为 L，它们可分别表示为：

$$u=[u_1,\ u_2,\ \cdots,\ u_L]^{\mathrm{T}}$$
$$y=[y_1,\ y_2,\ \cdots,\ y_L]^{\mathrm{T}}$$

现在要描述各个环节之间的连接关系。其实质就是把每个环节的输入信号 u_i 的来源以矩阵形式表达清楚，很明显，这些输入一是来自系统的外部信号 r；二是来自内部其他环节的输出 y_j，因而可写成

$$u=Pr+qy\tag{8.5}$$

其中 $r=[r_1,\ r_2,\ \cdots,\ r_{nr}]$ 是长度为 nr 的输入向量。故系数矩阵为 $P\,(L\times nr)$，$q\ (L\times L)$。P 称为输入矩阵，q 称为联接矩阵，P，q 的各元素只能取-1，1，0 三个值中之一，-1 表示负联接(或负反馈)，1 表示正联接(正反馈)，0 表示不联接。联接方程通常比较简单，可以用列出 u_i 的方程组求得，也可以直接写出。

将方程（8.5）代入方程（8.3），得到

$$\begin{aligned}\dot{x}&=\overline{A}x+\overline{B}Pr+\overline{B}qy\\ y&=\overline{C}x+\overline{D}Pr+\overline{D}qy\end{aligned}\tag{8.6}$$

把这个联立方程变换成标准的状态方程

$$\begin{aligned}\dot{x}&=Ax+Br\\ y&=Cx+Dr\end{aligned}\tag{8.7}$$

其中

$$\begin{aligned}A&=\overline{A}+\overline{B}q(I-\overline{D}q)^{-1}\overline{C}\\ B&=\overline{B}[I+q(I-\overline{D}q)^{-1}\overline{D}]P\\ C&=(I-\overline{D}q)^{-1}\overline{C}\\ D&=(I-\overline{D}q)^{-1}\overline{D}P\end{aligned}\tag{8.8}$$

对大多数物理系统，D_i=0，故 $\overline{D}$ =0，它意味着传递函数分母的阶数高于分子的阶数，这时公式（8.8）成为

$$A=\overline{A}+\overline{B}q\overline{C},\ B=\overline{B}P,\ C=\overline{C},\ D=0\tag{8.9}$$

公式（8.7）就是组合后系统的状态方程，其系数矩阵为 $A(n\times n)$，$B(n\times nr)$，$C(L\times n)$，$D(L\times nr)$。整个系统有 nr 个输入及 L 个输出。如果只要其中某 s 个输出。只要简单地删去 C 和 D 中用不着的各行，保留用得着的那 s 行，最后构成 $C(s\times n)$和 $D(s\times nr)$即可，这相当

于在系统的输出端再串一个输出矩阵，并在其中去除了无用的输出项。

系数公式（8.8），特别是公式（8.9）具有相当简洁的形式，但是，它的阶数很大，如果人工计算，无疑是十分冗繁和容易出错的，而用计算机辅助时，就变得非常简便了，特别是把环节用 LTI 对象表示时。采用集成的软件包，这些转换也都可以让机器自动去完成。人们只要输入各环节的 LTI 模型，再输入相应的联接矩阵和输入矩阵，并指定输出变量，软件包会自动判别输入的模型表述方式，作出相应的运算并最后给出组合后系统的状态方程。

MATLAB 求复杂系统任意组合的状态方程可以通过五个步骤来完成。

（1）对方框图中的各个环节进行编号，建立它们的对象模型。在有多输入多输出环节时对输入和输出也要按环节的次序分别进行编号，当然它们的编号会大于环节的编号。

（2）建立无连接的状态空间模型，append 命令可完成这个功能。

$$Sap = append(s1, s2, \cdots, sL)$$

（3）写出系统的联模矩阵 $\boldsymbol{Q}$

MATLAB 中为联接矩阵 $\boldsymbol{Q}$ 规定的形式与（8.5）式中的 $\boldsymbol{q}$ 略有不同。$\boldsymbol{q}$ 是一个元素取值为（-1, 0, 1）的 $n\times n$ 阶方阵，而 $\boldsymbol{Q}$ 则是只标注 $\boldsymbol{q}$ 中的非零项的一个矩阵。它的第一列是输入的编号，其后是连接该输入的输出编号，如果是负联接，这个元素的前面应加上负号。联接矩阵 $\boldsymbol{q}$ 与 $\boldsymbol{Q}$ 的关系举例如下。

$$\boldsymbol{q}=\begin{bmatrix}0 & -1 & 1 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}\qquad \text{对应于}\qquad \boldsymbol{Q}=\begin{bmatrix}1 & -2 & 3\\2 & 0 & 0\\3 & 4 & 0\\4 & 0 & 3\end{bmatrix}$$

$\boldsymbol{Q}$ 中的第二行可以列出，也可以省略。可见，两个矩阵都能表达同样的内容。在多数情况下，后一种方法键入的字符要少一些。

（4）选择组合系统中需保留的对外的输入和输出端的编号并列出。

```
inputs = [i1, i2, …]        outputs = [j1, j2, …]
```

（5）用 connect 命令生成组合后的系统。

```
sys = connect(sap, Q, inputs, outputs)
```

不管各个环节使用的是什么类型的对象，合成的结果都将是状态空间模型。例 8.4 将给出一个数字实例。

8.1.6　连续系统和采样系统之间的变换

随着在控制系统中愈来愈广泛地使用计算机，采样系统的分析设计也变得更加普遍和重要。在这类系统中，通常被控对象是物理世界中的连续系统，在控制器中采用了数字计算机。通过传感器测量出被控对象的状态，经过模拟／数字转换（A／D 变换），按照一定的采样时间间隔，以数码的方式读入计算机；由计算机经过适当的数学和逻辑运算处理后，以数码的方式向执行器发送控制信号。因为按冯·诺依曼方式工作的计算机，它的 CPU 每瞬时只能执行一条命令，因此它的输出信息的周期至少应等于信息处理所需的时间。

这个数码形式的控制量，通常要经过数字／模拟转换（D／A 变换），才能与执行器相匹配。即使有些执行器本身是数字式的，比如同步卫星轨迹和姿态控制所用的脉冲式火箭发动机，或者是步进马达等，但它们的输出最终仍表现为能影响被控状态的连续变量。因此，几

乎没有任何一个采样控制系统能完全用差分方程或 z 变换来表示。它们的典型构成方式是兼有连续系统部分和采样控制部分，在相互联接的地方，是 D／A 转换和 A／D 转换。A／D 转换是一采样器，它测出采样瞬间的状态变量送给计算机去处理；而 D／A 变换器则通常是一个采样保持器，把某一瞬间计算机的控制命令变为电压后，一直保持到下一个数据到来为止。

对这样一类由连续部分和采样离散部分混合构成的系统，必须为它建立统一的描述模型，才能调用 MATLAB 中的相应工具，由于采样系统方程比较容易求解，为了实现快速仿真，使系统仿真尽量接近实时运行，人们往往把连续系统有意地转化为性能相当的采样系统；反过来，有时人们用测量和辨识的方法，得到系统差分方程模型，希望由它求得相应于实际物理世界的连续系统模型，这也是上面的逆问题，解决了这些问题，6.4 节中的模型描述表 6.1 中的左右两列之间的相互转换也就解决了。

连续系统到采样系统的转换关系如下，对于状态方程为

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu}$$
$$\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du}$$

的连续系统，对应的采样系统状态方程为

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{\mathrm{d}}\boldsymbol{x}(k) + \boldsymbol{B}_{\mathrm{d}}\boldsymbol{u}(k)$$
$$\boldsymbol{y}(k) = \boldsymbol{C}_{\mathrm{d}}\boldsymbol{x}(k) + \boldsymbol{D}_{\mathrm{d}}\boldsymbol{u}(k)$$

其中

$$\boldsymbol{A}_{\mathrm{d}} = \mathrm{e}^{\boldsymbol{A}t},\ \ \boldsymbol{B}_{\mathrm{d}} = \int_0^{T_s} \mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\mathrm{d}\tau, \boldsymbol{C}_{\mathrm{d}} = \boldsymbol{C},\ \ \boldsymbol{D}_{\mathrm{d}} = \boldsymbol{D}$$

T_s 为采样周期。

反之，采样系统到连续系统的转换关系为上式的逆

$$\boldsymbol{A} = \frac{1}{T_s}\ln(\boldsymbol{A}_{\mathrm{d}}),\ \ \boldsymbol{B} = (\boldsymbol{A}_{\mathrm{d}} - \boldsymbol{I})^{-1}\boldsymbol{A}\boldsymbol{B}_{\mathrm{d}},\ \ \boldsymbol{C} = \boldsymbol{C}_{\mathrm{d}},\ \ \boldsymbol{D} = \boldsymbol{D}_{\mathrm{d}}$$

需要指出的是，虽然算式挺简明，但因为这些系数都是矩阵，连续系统与采样系统之间的转换计算是十分繁杂的，即使是三阶系统，用手工进行运算也是非常困难的。因此，计算辅助设计在这个领域就更显得不可缺少。MATLAB 控制工具箱提供了三种功能很强的函数来完成这个使命。它们是：c2d(连续系统变为采样系统)、d2c(采样系统变为连续系统)和 d2d(采样系统改变采样频率)。旧版本控制工具箱中的 c2dm 和 d2cm 等函数都已列入待作废的函数库。

■ c2d 函数的调用格式为

➢ sd = c2d(sc, Ts)　　把连续系统以采样周期 Ts 和零阶保持器方式转换为采样系统。

➢ sd = c2d(sc, Ts, method)　　把连续系统以采样周期 Ts 和 method 方法，转换为采样系统。

method（方法）有五种，对应下列字符串，在编程调用时只需键入第一个字符。

- zoh——零阶保持器（可缺省）
- foh——一阶保持器
- tustin—— 双线性变换（tustin）法
- prewarp——频率预修正双线性变换法，用此法时还增加一个变元（边缘频率 wc）
- matched——根匹配法

■ d2c 是 c2d 的逆运算，其调用格式与 c2d 相仿，只是 Ts 已包含在模型属性中，无需

再作为变元输入，另外，method 中没有 foh 选项。

➢ sc=d2c(sd, method)　　把采样系统以 method 方法，转换为连续系统。

■ d2d 函数的调用格式为

➢ sd2 = d2d(sd1, Ts2)　　把采样系统 1 的原采样周期 Ts1 改为 Ts2，转换为采样系统 2。其实际的变换过程是，先把待变换的采样系统按零阶保持器转换为原来的连续系统，然后再用新的采样频率和零阶保持器转换为新的采样系统。例 8.5 将给出数字实例。

8.1.7　典型系统的生成

用表 8.5 列出的函数可以快速地生成所需阶数的线性时不变系统。

表 8.5　生成线性时不变系统的函数

函数名称及典型调用方式	功　　能
s = rss(n)	随机生成 *n* 阶稳定的连续状态空间模型
[num, den] = rmodel(n)	随机生成 *n* 阶稳定的连续线性模型系数
s = drss(n)	随机生成 *n* 阶稳定的离散状态空间模型
[num, den] = drmodel(n)	随机生成 *n* 阶稳定的离散线性模型系数
[num, den] = ord2(wn, z)	生成固有频率为 wn 阻尼系数为 z 的二阶系统系数

例如键入

```
sys=rss(4)
```

得出一个随机产生但却是稳定的状态空间系统 sys，其系数矩阵为

```
a =                  x1          x2          x3          x4
           x1    -1.32676    -0.22835     0.15692     0.13866
           x2    -0.22835    -1.43743     0.86047     0.11163
           x3     0.15692     0.86047    -1.36414    -0 60837
           x4     0.13866     0.11163    -0.60837    -1.30948
b =                  u1
           x1     0
           x2    -0.51864
           x3     0.32737
           x4     0.23406
c =                  x1          x2          x3          x4
           y1     0.02147    -1.00394    -0.94715    -0.37443
d =                  u1
           y1        0
Continuous-time system.
```

如果键入

```
sys=rss(4, 3, 2)
```

就得出一个四阶的双输入三输出的稳定的状态空间系统，读者可自行检验。

■ rmodel 函数用于产生 LTI 对象的系数，它并不生成 LTI 对象本身，它的左端放几个输出变量就决定了要几个系数矩阵，也决定了生成的 LTI 对象的类型。例如键入

```
[num, den] = rmodel(4)
```

得到传递函数模型的系数

```
num =    [ 0          0         0          0.5354     0.3360 ]
den =    [ 1.0000    10.7257    38.0286    38.8216    10.1723 ]
```

键入

```
[a, b, c, d] = rmodel(4)
```

就得到状态空间模型的系数

```
a =
   -0.3773    0.1471    0.0180   -0.0606
    0.1471   -0.7013    0.0528    0.2233
    0.0180    0.0528   -2.3237   -1.3597
   -0.0606    0.2233   -1.3597   -1.3350
b =     0.3792
        0.9442
       -2.1204
       -0.6447
c =    -0.7043   -1.0181   -0.1821    1.5210
d =    -0.0384
```

在 rmodel 函数中再增加两个输入变元，成为 rmodel(n，p，m)，同样可以产生 m 输入 p 输出的系统系数矩阵。

drss 和 drmodel 的用法相仿，不同点仅仅在于它生成的是离散系统。

ord2 函数也是用来产生二阶系统的系数的，不能生成系统本身，因此，它的左端输出变量的数目为四个或两个，决定了生成的系统属于状态空间还是传递函数类型，生成的传递函数为

$$H(s)=\frac{1}{s^2+2\zeta\omega_n s+\omega_n{}^2}$$

键入　`[num, den]=ord (10, 0.5)`

结果为　`num=1, den=[1, 10, 100]`

键入　`[a, b, c, d]=ord2(10, 0.5)`

结果为

```
a =     0       1
      -100     -10
b =     0
        1
c =     1       0
d =     0
```

【例 8.1】 SIMO 系统几种模型转换方法的比较

已知系统的动态特性由下列状态空间模型描述

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}1 & -1 & 0\\ 0 & 2 & 0\\ 1 & 0 & 4\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}1\\ 0\\ -1\end{bmatrix}u$$

$$\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}2 & 0 & 0\\ 1 & 2 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

求出它的传递函数模型、零极增益模型、极点留数模型。

解：按上述方程输入状态方程系数矩阵 ***A***，***B***，***C***，***D***

A＝[1，−1，0;0，2，0;1，0，4]；***B***＝[1；0；−1]；***C***＝[2，0，0；1，2，3]；***D***＝[0；0]；

注意这是一个单输入双输出系统，***D*** 是 2×1 阶的。故必须置为[0；0]。有多种方法可以用来解这个问题，为了便于做比较，用逐次键入命令的方法将这些语句集合在一起，就得到程序 q801.m。

■ **MATLAB程序q801.m**

```
A=[1, -1, 0; 0, 2, 0; 1, 0, 4]; B=[1; 0; -1]; C=[2, 0, 0; 1, 2, 3]; D=[0; 0];
[f, g]=ss2tf(A, B, C, D), pause
printsys(f, g, 's'), pause
 [z, p, k]=ss2zp(A, B, C, D), pause
sys=ss(A, B, C, D);, pause
[f1, g1]=tfdata(sys), pause                % 转换提取 tf 系数向量
f1{:, :}, g1{:, :}, pause                  % 提取 tf 系数向量具体值
[z1, p1, k1]=zpkdata(sys), pause           % 转换提取 zpk 系数向量
z1{:, :}, p1{:, :}, pause                  % 提取 zpk 系数向量具体值
systf=tf(sys), pause                       % 生成等价的 tf 对象的 LTI 模型
syszp=zpk(sys), pause                      % 生成等价的 zpk 对象的 LTI 模型
```

● 方法 1　　用旧的控制系统工具箱命令

键入　[f, g]=ss2tf(A, B, C, D)

```
f=   0     2   -12    16
     0    -2     6    -4
g=   1    -7    14    -8
```

写得明确些，这个单输入双输出系统有两个传递函数，表示如下。

$$H(s)=\frac{f(s)}{g(s)}=\begin{bmatrix}\dfrac{f(1,:)}{g}\\ \dfrac{f(2,:)}{g}\end{bmatrix}=\begin{bmatrix}\dfrac{2s^2-12s+16}{s^3-7s^2+14s-4}\\ \dfrac{-2s^2+6s-4}{s^3-7s^2+14s-8}\end{bmatrix}=\begin{bmatrix}\dfrac{Y_1(s)}{U(s)}\\ \dfrac{Y_2(s)}{U(s)}\end{bmatrix}$$

类似地可求得零极增益模型参数

```
[z, p, k]=ss2zp(A, B, C, D)
z  =  2      2
      4      1
```

```
p =   4
      1
      2
k  =  2.0000
     -2.0000
```

写成便于阅读的形式，即由如下两个传递函数组成一个2×1阶的传递函数矩阵。

$$H(s)=\begin{bmatrix}\dfrac{2(s-2)(s-4)}{(s-4)(s-1)(s-2)}\\ \dfrac{-2(s-1)(s-2)}{(s-4)(s-1)(s-2)}\end{bmatrix}=\begin{bmatrix}\dfrac{Y_1(s)}{U(s)}\\ \dfrac{Y_2(s)}{U(s)}\end{bmatrix}$$

用这种方法，只能得出系数向量的值，不能得出便于阅读的形式。而且必须知道输入输出变量的数目，才能编程，程序难以通用化。

● 方法2 用LTI对象和新的控制系统工具箱命令提取参数

键入

```
sys=ss(A, B, C, D);
[f1, g1]=tfdata(sys), pause          % 转换提取 tf 系数向量
```

得

```
f1 = [1×4 double]
     [1×4 double]
g1 = [1×4 double]
     [1×4 double]
```

要提取tf系数向量具体值，再键入花括号下标

```
f1{1, :}, g1{1, :},    f1{2, :}, g1{2, :}
```

得

```
ans =    0    2   -12    16
ans =    1   -7    14    -8
   ……
```

类似地可以转换提取zpk系数向量

```
[z1,p1,k1]=zpkdata(sys), z1{:, :}, p1{:, :}, pause
```

可以检验 f，g，z，p，k 与 f1，g1，z1，p1，k1 是完全相等的。方法2的好处是不必知道输入输出变量的数目，缺点是仍然得不到便于阅读的形式。

● 方法3 用LTI对象和新的控制系统工具箱命令再建新模型

键入

```
systf=tf(sys), pause              % 生成等价的 tf 对象的 LTI 模型
```

得

```
Transfer function from input to output...
        2 s^2 - 12 s + 16
 #1:  ---------------------
      s^3 - 7 s^2 + 14 s - 8
         -2 s^2 + 6 s - 4
 #2:  ---------------------
      s^3 - 7 s^2 + 14 s - 8
```

再键入

```
syszp=zpk(sys), pause        % 生成等价的 zpk 对象的 LTI 模型
```

得

```
Zero/pole/gain from input 1 to output...
         2 (s-2) (s-4)
#1:  ----------------
      (s-4) (s-2) (s-1)
        -2 (s-2) (s-1)
#2:  ----------------
      (s-4) (s-2) (s-1)
```

可见，方法 3 输入的程序量最小，而得出的结果最清楚，因此是最好的。

【例 8.2】 含串联和反馈环节的系统传递函数

列写图 8.1 所示系统的传递函数，分别设

（1）K1 = 250

（2）K1 = 1000

并求系统的传递函数和极点分布。

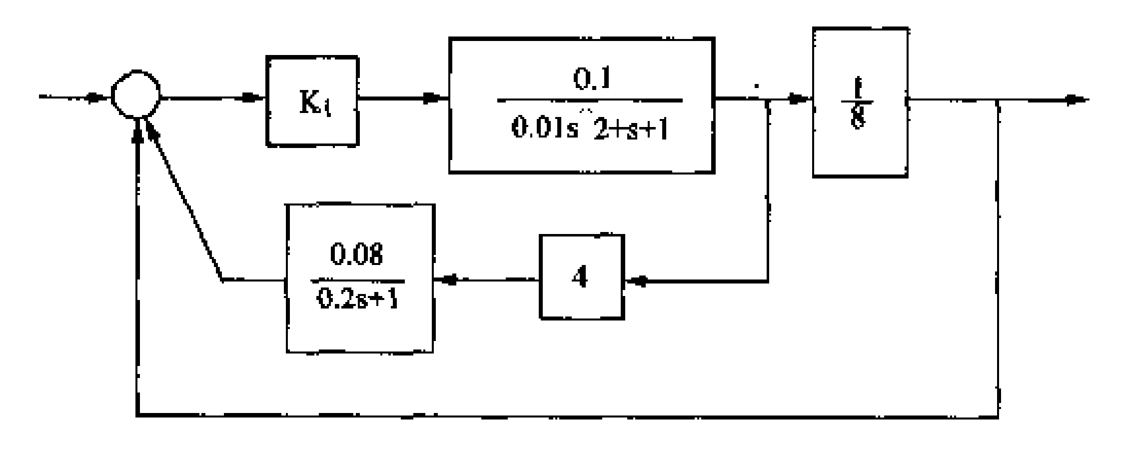


图 8.1 例 8.2 的系统框图

解：■ 建模

● 方法 1 用旧的控制系统工具箱命令设 Wb1 为内环的闭环传递函数，fb1 为其分子系数向量，gb1 为其分母系数向量，则有

```
[fb1, gb1]=feedback(0.1*K1, [0.01, 1, 1], 4*K2, [0.2, 1])
```

设 Wb 为全系统（即外环）的闭环传递函数，fb，gb 分别为其分子分母系数向量，则有

```
[fb, gb]=cloop(fb1, conv(gb1, [1, 0]))
```

对不同的 K1，可以设置一个 for 循环来完成计算。

● 方法 2 用新的控制系统工具箱命令和 LTI 对象。

■ **MATLAB程序q802.m**

```
% 方法 1  用旧的控制系统工具箱令令
K2=0.08
for K1=[250, 1000]
   [fb1, gb1]=feedback(0.1*K1, [0.01, 1, 1], 4*K2, [0.2, 1]);
   [fb, gb]=cloop(fb1, conv(gb1, [1, 0]));
   K1, K2, printsys(fb, gb)
   p=roots(gb)'
end
```

```
% 方法 2  用新的控制系统工具箱命令和LTI 对象，为节约篇幅，设定 K1 和 K2
K1=250; K2=0.008
s1=tf(0.1*K1, [0.01, 1, 1]);      % 建立环节的 LTI 对象模型
s2=tf(4*K2, [0.2, 1]);
s3=zpk([ ], 0, 1);
sb1= feedback(s1, s2);            % 对内环应用反馈公式
s = feedback(series(sb1, s3), 1)  % 对外环再用反馈公式，反馈环节系统函数为 1
```

■ 方法 1 的程序运行结果

```
K1 =    250,    K2 =  0.0800
num/den =               5 s + 25
          ------------------------------------------
            0.002 s^4 + 0.21 s^3 + 1.2 s^2 + 14 s +25
p=[-99.6723; -1.6574 +7.7170i; -1.6574 -7.7170i; -2.0130 ]
K1 =    1000, K2 =   0.0800
num/den =               20 s + 100
     ------------------------------------------------------
   0.002 s^4  +  0.21 s^3  +  1.2 s^2  +  53 s  +  100
p =[  -101.61; -72 + 15.89i; -0.72 - 15.89i; -1.94 ]
```

■ 方法 2 程序运行结果

```
Zero/pole/gain:
                   2500 (s + 5)
------------------------------------------------------------
(s + 99.67)  (s + 2.013)  (s^2  +  3.315s  +  62.3)
```

根据控制系统工具箱 LTI 对象运算优先等级为“状态空间>零极增益>传递函数”的规定，合成系统的系统函数的对象特性应按照环节的最高等级来确定。在本章的例子中，有一个环节使用零极增益，其他两个是传递函数，因此，最后的系统函数就表现为零极增益。可以看出，它的极点与方法 1 中的第一个结果相同。

【例 8.3】 用信号流图和 LTI 对象解复杂系统

设系统的信号流图如图 8.2 所示，其中

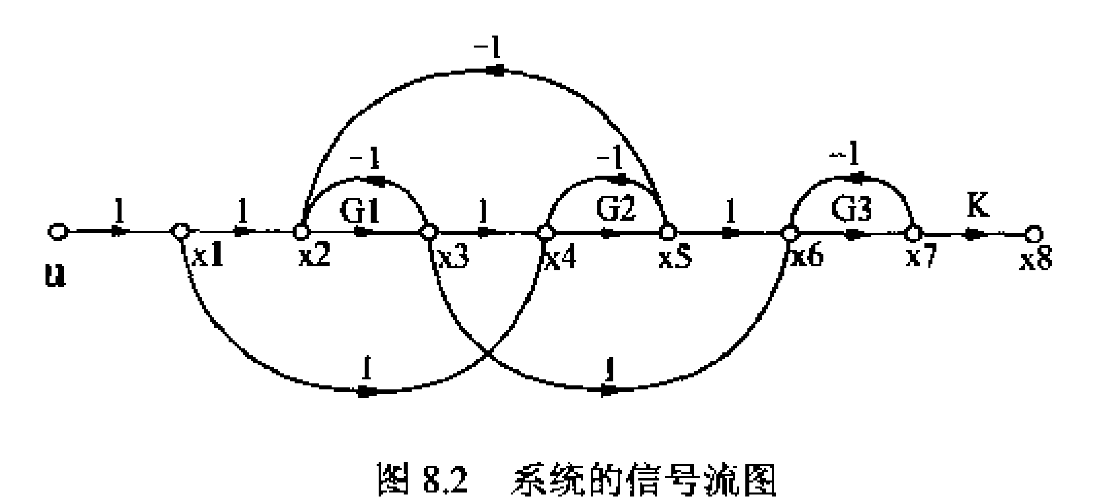


图 8.2　系统的信号流图

$$G_1 = \frac{s}{s+1}$$

$$G_2 = \frac{3}{s+2}$$

$$G_3 = \frac{s+4}{s^2+5s+6}$$

求以 u 为输入，x_8 为输出的系统函数。

解：■ 建模

信号流图的建模方法不再重复，为了得出不同的对象类型，可以在输入环节中重新安排。按照运算的优先级，只要有一个环节用 ss 对象，结果必是 ss 对象；如果每个环节都不

用 ss 对象，只要有一个环节用 zpk 对象，结果必是 zpk 对象；只有所有环节都用 tf 对象，结果才是 tf 对象。但这里还有一个要点，就是矩阵 R 的对象类型取决于其第一个赋值元素的对象类型。在本例中，第一个赋值语句是 R(3, 2) = G1；因此，G1 的对象类型就决定了 R 的对象类型。为了验证这一点，在程序中设置了可选项。程序的其他部分同例 6.19。

■ MATLAB程序q803.m

```
clear,
k=input('用什么模型? 传递函数-键入 1, 零极增益-键入 2, 状态空间-键入 3 ', k= );
switch k
case 1
   G1=tf([1, 0], [1, 1]),          % 定义 lti 对象, 全用 tf 形式
   G2=tf(3, [1, 2]),
   G3=tf([1, 4], [1, 5, 6])
case 2
   G1=zpk(0, -1, 1),               % 定义 lti 对象, 有两个用 zpk 形式
   G2=zpk([ ], -2, 3),
   G3=tf([1, 4], [1, 5, 6])
otherwise
   G1=ss(zpk(0, -1, 1)),           % 定义 lti 对象, 有两个用 zpk 形式
   G2=zpk([ ], -2, 3),
   G3=tf([1, 4], [1, 5, 6])
end
R(3, 2)=G1;                        % 采用字符矩阵, 第一条赋值语句右端必须是字符变量
R(2, 1)=1;R(2, 3)=-1;R(2, 5)=-1;   % 列出连接矩阵
R(4, 3)=1;R(4, 1)=1;R(4, 5)=-1;
R(5, 4)=G2;
R(6, 3)=1;R(6, 5)=1;R(6, 7)=-1;
R(7, 6)=G3;
R(8, 7)=5;
R(:, 8)=zeros(8, 1);           % 加一个全零列, 补成方阵
P=[1; 0; 0; 0; 0; 0; 0; 0];
I=eye(size(R));
W=(I-R)\P;                     % 求出完整的传递矩阵
if k==3 W,                     % 若要状态空间, 输出 W
else W8=W(8),                  % 若非状态空间, 输出 x8 的传递函数为 W 中的第 8 行
end
```

■ 程序运行结果

键入 k=1 时，得出的 W8

```
Transfer function:
 2.5 s^3 + 37.5 s^2 + 117.5 s + 30
-----------------------------------------
s^4 + 13 s^3 + 54.5 s^2 + 85 s + 25
```

键入 k=2 时，得出的 W8

```
Zero/pole/gain:
  2.5 (s+10.72) (s+4) (s+0.2798)
---------------------------------------------
(s+6.622) (s+0.3775) (s^2 + 6s + 10)
```

键入 k=3 时，得出的 W

```
a =        x1        x2         x3         x4
           x1   -0.50000   -0.86603   0          0
           x2   -0.86603   -6.50000   0          0
           x3   -0.50000    0.86603   -6.00000   -2.50000
           x4    0          0          4.00000    0
b =                u1
           x1    0.50000
           x2    2.59808
           x3    0.50000
           x4          0
c =              x1         x2         x3         x4
           y1    0          0          0          0
           y2    0.50000   -0.86603    0          0
           y3   -0.50000   -0.86603    0          0
           y4   -0.50000   -2.59808    0          0
           y5    0          1.73205    0          0
           y6   -0.50000    0.86603   -1.00000   -1.00000
           y7    0          0          1.00000    1.00000
           y8    0          0          5.00000    5.00000
d =                u1
           y1    1.00000
           y2    0.50000
           y3    0.50000
           y4    1.50000
           y5    0
           y6    0.50000
           y7    0
           y8    0
Continuous-time System.
```

可以看出，在用状态空间模型时，W8 是没有意义的，只能有 W，它用 a，b，c，d 表述。要求出对 y8 的状态空间系数矩阵，可以取 a，b，c(8, :)和 d(8, :)来组成。

【例 8.4】 复杂框图的结构图变换

由图 8.3，设

$$A=\begin{bmatrix}-9 & 17\\ -2 & 3\end{bmatrix},\quad B=\begin{bmatrix}-0.5 & 0.5\\ -0.002 & -1.8\end{bmatrix},$$

$$C=\begin{bmatrix}-3 & 2\\ -13 & 18\end{bmatrix},\quad D=\begin{bmatrix}-0.5 & -0.1\\ -0.6 & 0.3\end{bmatrix}$$

求合成系统的系统模型。

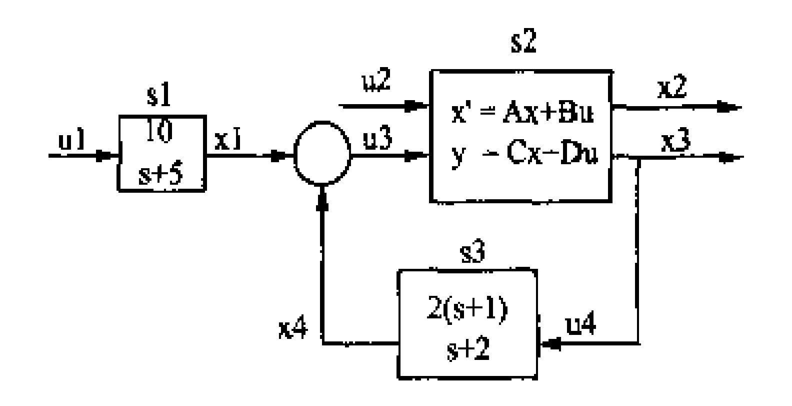


图 8.3　例 8.4 的框图

解：■ 建模

按下述的步骤来做。

对方框图中的各个环节及其输入和输出进行编号，建立它们的对象模型。

环节编号　s1，s2（两输入两输出），s3

输入端编号　u1，u2，u3，u4

输出端编号　x1，x2，x3，x4

建立无联接的状态空间模型，可用 append 命令完成这个功能。

➢ sap=append(s1，s2，3)

写出系统的联接矩阵 $\boldsymbol{Q}$

$$Q=\begin{bmatrix}1 & 0 & 0\\ 2 & 0 & 0\\ 3 & 2 & -4\\ 4 & 3 & 0\end{bmatrix}$$

$\boldsymbol{Q}$ 中的第一行第二行可以省略。

选择组合系统中需保留的对外的输入和输出端的编号并列出。

inputs=[1，2]　　outputs=[2，3]

用 connect 命令生成组合后的系统。

sys=connect(sap，Q，inputs，outputs)

■ MATLAB程序q804.m

```
% 建立环节的对象模型
s1 = tf(10, [1, 5], 'inputname', 'u1', 'outputname', 'x1');
A=[-9, 17;-2, 3];
B=[-0.5, 0.5;-0.002, -1.8];
C=[-3, 2;-13, 18];
D=[-0.5, -0.1;-0.6, 0.3];
s2 = ss(A, B, C, D, 'inputname', {'u2', 'u3'}, 'outputname', {'x2', 'x3'});
s3 = zpk(-1, -2, 2, 'inputname', 'u4', 'outputname', 'x4');
sap = append(s1, s2, s3);          % 建立无连接的总状态空间模型
Q=[3, 2, -4;4, 3, 0];              % 写出系统的联接矩阵 Q
% 确定外输入输出。系统的外输入为 r=[u1, u2]，对外输出为 y=[x2, x3]
inputs=[1, 2]; outputs = [2, 3];
% 用 connect 命令生成组合后的系统。
sc = connect(sap, Q, inputs, outputs);
set(sc, 'inputname', ['r1';'r2'], 'outputname', {'y1';'y2'})
sc                       % 显示合成系统特性
```

■ 程序运行结果

可以键入 s1，s2，s3，sap 等来获得各阶段各子系统特性，这里只显示了合成系统特性 sc。

```
a =            x1          x2          x3          x4
       x1    -5.00000      0           0           0
       x2     0           -2.23529     7.00000     0.41595
       x3     0          -26.35294    39.00000    -1.49740
       x4     0          -12.64473    16.97056    -1.64706
b =            r1          r2
       x1     4.00000      0
       x2     0           -0.29412
       x3     0           -0.74318
       x4     0           -0.67383
c =            x1          x2          x3          x4
       y1     0           -4.35294     4.00000    -0.08319
       y2     0           -8.94118    12.00000     0.24957
d =            r1          r2
       y1     0           -0.54118
       y2     0           -0.47647
Continuous-time system.
```

【例 8.5】 连续系统变换为离散系统

已知连续系统的传递函数为

$$H(s)=\frac{-4s+1}{s^2+2s+10}$$

采样周期为 0.2s，试用零阶保持器和双线性变换两种方法求出其离散传递函数。

解：■ 建模

本例所用建模方法可参阅 8.1.6 节的内容，这里不再赘述。

■ MATLAB程序q805.m

```
format compact
f=[-4, 1];g=[1, 7, 12];ts=0.2;
sc=tf(f, g)                 % 建立连续系统对象模型
disp('零阶保持器')           % 用不同的方法变换为采样系统
sd1=c2d(sc, ts)             % 零阶保持器方法
disp('双线性变换')
sd3=c2d(sc, ts, 't')        % 双线性变换方法
```

■ 程序运行结果

零阶保持器

```
Transfer function:
  -0.3852 z + 0.4059
------------------------------
z^2 - 0.9981 z + 0.2466
Sampling time: 0.2
```

双线性变换

```
Transfer function:
 0.2143 z^2 + 0.01099 z + 0.2253
---------------------------------
    z^2 - 0.967 z + 0.2308
Sampling time: 0.2
```

8.2　动态特性和时域分析函数

控制工具箱中的动态特性和时域分析函数参看表 8.6。

表 8.6　控制工具箱中的动态特性和时域分析函数

参数名称和典型调用格式		功　能
零极点分析及极轨迹绘制函数	[wn, zeta]=damp(sys)	系统极点的固有频率 wn 和阻尼系数 zeta
	k=dcgain(sys)	直流（稳态）增益
	ps=dsort(p)	离散系统极点按幅值降序排列
	ps=esort(p)	连续系统极点按实部降序排列
	p=pole(sys)	系统的极点计算
	[v, p]=eig(sys)	系统的特征根 p 和特征向量 v 的计算
零极点分析及根轨迹绘制函数	pzmap(sys) [p, z]=pzmap(sys)	系统零极点绘制
	z = tzero(sys)	求系统的传输零点
	rlocfind(sys)	按鼠标选定根轨上的点计算其增益和极的值
	rlocus(sys)	计算和绘制系统的根轨迹
	sgrid	绘制连续系统根平面上的等阻尼和等固有频率网格
	zgrid	绘制离散系统极平面上的等阻尼和等固有频率网格
时域动态分析函数	[u, t]=gensig(type，tau)	生成特定类型（方波、正弦等）的激励信号
	impulse(sys) [y, t, x]=impulse(sys)	计算和绘制系统的脉冲响应
	initial(sys，x0) [y, t]=initial(sys, x0)	计算和绘制系统的零输入响应
	[y, t] =lsim(sys, u)	计算系统在任意输入作用下的输出响应
	step(sys) [y, t, x]=step(sys)	计算和绘制系统的阶跃响应
	[P, Q]=covar(sys, w)	白噪声激励下系统输出和状态协方差 P，Q 的计算

这些函数中凡是以系统名称 sys 作为输入变元的，都同时适用于连续系统和离散系统。而且也适用于多输入多输出系统。因为这些特征都已包含在系统名称中。

■ 与系统的零极点有关的有以下几个函数，它们都可以用来判断系统的稳定性。

➢ p=pole(sys)　　用来计算系统的极点。如果系统有重极点，计算结果不一定准确。

如果 sys 是采样系统，则获得的是映射到连续系统 s 平面上的等效极点。如果采样周期无定义，则返回值为空。p 的实部为负时，系统稳定。

➢ p = eig(sys)　　与 p = pole(sys)相同。

eig 函数功能要多一些，如输入[v, p] = eig(sys)，可以同时得到系统的特征根 p 和特征向量 v。p 是以对角矩阵的形式返回的。

➢ pr = esort(p)　　系统极点按实部降序排列，通常用于连续系统分析。

稳定的连续系统极点实部为负，故实部绝对值较小的根，也即影响过渡过程时间的主导根排在前。

➢ pa = dsort(p)　　系统极点按幅值降序排列，通常用于离散系统分析。

稳定的离散系统极点幅值小于 1，故幅值接近于 1 的根，即影响过渡过程时间的主要根排在前。

➢ [wn，zeta]=damp(sys)　　用来计算系统所有共轭极点的固有频率 wn 和阻尼系数 zeta。

如果 sys 是采样系统，则获得的是映射到连续系统 s 平面上的等效极点。如果采样周期无定义，则返回值为空。

damp 函数的算法核心是 p=roots(poly(A))、wn=abs(p)、zeta= −real(p)./wn。zeta 为正时，系统稳定。

➢ k=dcgain(sys)　　计算直流（稳态）增益。

dcgain 函数的算法公式为，对连续系统，$\boldsymbol{K}=\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$；对离散系统，$\boldsymbol{K}=\boldsymbol{D}-\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}$。

如果 sys 是多输入多输出系统，例如二输入三输出系统，则得出的 k 将是 3×2 阶矩阵。

➢ z = tzero(sys)　　求系统的传输零点，适用于连续和离散系统，也适用于多输入多输出系统。

➢ [z, gain] = tzero(sys)　　只适用于单输入单输出系统，注意这个 gain 不是直流增益，而是 zpk 对象中的 k。如 sys 是多输入多输出系统，返回的 gain 为空。

➢ pzmap(sys)　　不带左端输出变量时，用来计算和绘制系统零极点。

对连续系统，在 s 平面上绘制；对离散系统，在 z 平面上绘制。结果都不返回数据，只返回图形。

带左端输出变量的调用格式[p, z] = pzmap(sys) 用来计算并返回零极点，不返回图形。

➢ sgrid　　绘制连续系统根平面（s 平面）上的等阻尼和等固有频率网格，见图 8.4-1。

➢ zgrid　　绘制离散系统根平面（z 平面）上的等阻尼和等固有频率网格，见图 8.4-2。

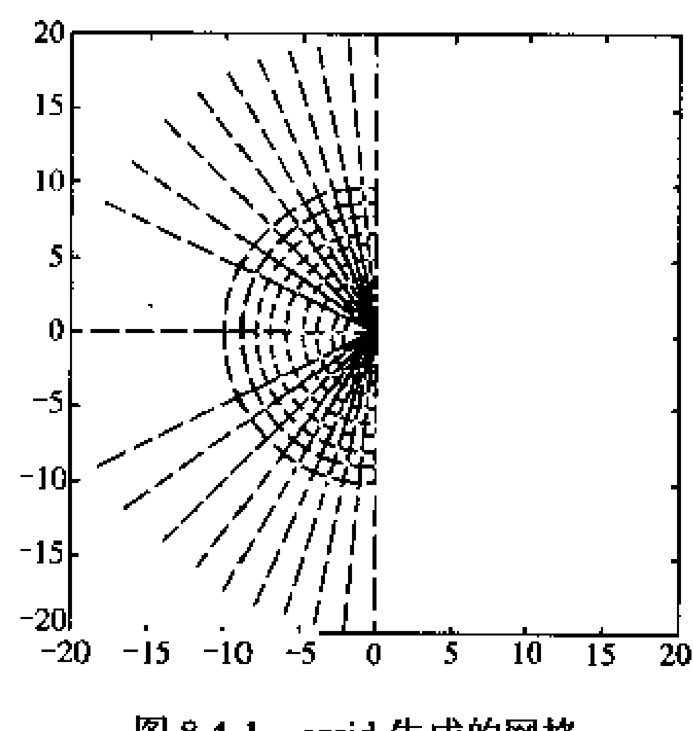


图 8.4-1　sgrid 生成的网格

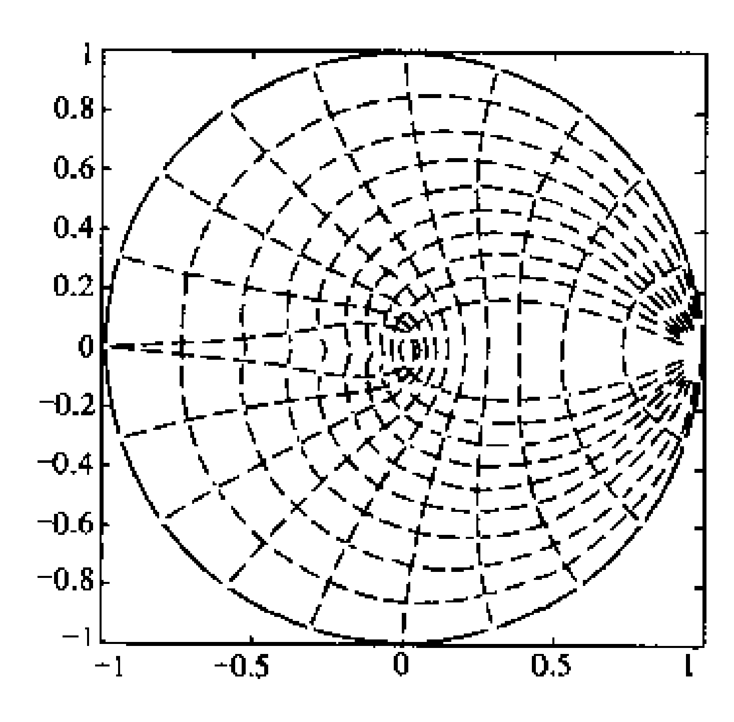


图 8.4-2　zgrid 生成的网格

➢ rlocus(sys)　计算和绘制系统的根轨迹图（开环增益 k 从零到无穷大）。

➢ rlocus(sys, k)　按给定的开环增益数组 k 的范围计算和绘制系统的根轨迹。

➢ [r, k] = rlocus(sys)　计算系统的根轨迹，返回根 r 及相应 k 的数组值，不返回根轨迹图形。

➢ rlocfind(sys)　先由 rlocus 函数画出系统的根轨迹图，再键入 rlocfind(sys)，根轨迹图上会出现随鼠标移动的十字线，用鼠标左键选定该根轨上的点，MATLAB 将计算并显示其增益和根的值。本函数同时适合于连续和离散系统。

➢ [k, r] = rlocfind(sys)　把找到的增益和根的值分别赋值给变量 k 和 r。

■ 与系统的时域分析有关的有以下函数。

➢ impulse(sys)　不带左端输出变量时，它绘制系统 sys 的脉冲响应，结果不返回数据，只返回图形。

如果是多输入多输出系统，画出的脉冲响应将自动分成相应分割的子图。

带左端输出变量的调用格式[y, t, x] = impulse(sys)，用来计算系统的脉冲响应，给出系统输出变量和状态变量随时间变化的数值解，不返回图形。

impulse(sys, t) 的第二个变元 t 若为标量，表示终止时间；若为数组，表示需计算的时刻。

impulse (sys1, sys2, …, sysN, t) 可以在一张图上画出多个系统的脉冲响应。

impulse (sys1, 'PlotStyle1', sys2, 'PlotStyle2', …, t)可以规定多个系统的脉冲响应的线型。

➢ initial(sys, x0)　不带左端输出变量时，用来计算和绘制系统 sys 在初始条件 x0 下的零输入响应，不返回数据，只返回图形。

格式 initial(sys, x0, t)的第三个变元 t 若为标量，表示终止时间；若为数组，表示需计算的时刻值。

带左端输出变量的调用格式[y, t] = initial(sys, x0)，用来计算系统在初始条件 x0 下的零输入响应，它给出输出变量和状态变量随时间变化的数值解，结果不返回图形。其他调用格式与 impulse 同。

➢ step(sys)　不带左端输出变量时，计算和绘制系统的阶跃响应，结果不返回数据，只返回图形。

格式 step(sys, t)的第二个变元 t 若为标量，表示终止时间；若为数组，表示需计算的时刻值。

带左端输出变量的调用格式[y, t, x] = step(sys)，用来计算系统的阶跃响应，给出系统输出变量和状态变量随时间变化的数值解，不返回图形。其他调用格式与 impulse 同。

➢ lsim(sys, u, t)　不带左端输出变量的格式用来计算和绘制 sys 在初始条件 x0 和任意输入 u 作用下的输出 y，不返回数据，只返回图形。与其他几个命令不同的是用 lsim 函数时，必须给出时间数组 t，对离散系统，时间数组 t 的步长必须与采样周期 Ts 相同。

带左端输出变量的调用格式[y, t] = lsim(sys, u, t, x0)，用来计算系统 sys 在初始条件 x0 和任意输入 u 作用下的输出 y 的数值解，不返回图形。其他调用格式与 impulse 同。

impulse，initial，step，lsim 的计算原理和编程方法，在本书第 5 章和第 6 章的许多例题中都已给出。

➢ [u, t] = gensig(type, tau)　用来生成特定类型（方波、正弦等）的激励信号 u。

变元 type 可取字符串 sin(正弦)，square(方波)，impulse(脉冲)之一。Tau 为信号周期。

格式[u, t] = gensig(type, tauTf, Ts) 中，增加的变元 Tf 为信号持续时间，Ts 为采样周期。

➢ [P, Q] = covar(sys, w)　　用于白噪声 w 激励下系统输出协方差 P 和状态协方差 Q 的计算。要求输入为满足下式的高斯白噪声。

$$E(\omega(t)\omega(\tau)^{T}) = W\delta(t-\tau) \qquad \text{（连续系统）}$$

$$E(\omega(k)\omega(i)^{T}) = W\delta \qquad \text{（离散系统）}$$

稳态输出协方差 P 和状态协方差 Q 的定义为

$$P = E(yy^{T}) \ , \qquad Q = E(xx^{T})$$

【例 8.6】 阻尼系数对二阶系统脉冲响应的影响

二阶系统的传递函数为

$$H(s) = \frac{1}{s^{2} + 2\zeta\,\omega_{n}s + \omega_{n}^{2}}$$

设其固有频率ω_n=10，在阻尼系数ζ = [0.1，0.3，0.7，1]时，分别画出其脉冲响应函数。将系统在条件 T_s = 0.1 下离散化，同样画脉冲响应函数曲线。

解：■ 建模

先用 ord2 函数建立二阶连续系统 LTI 模型 s，用 c2d 函数转换为离散 LTI 模型 sd，再用 impulse 函数绘制脉冲响应曲线。不同的 zeta=ζ 用 for 循环处理。注意 impulse(sys)函数对连续系统和离散系统是公用的，它会根据 sys 的不同属性自动选择相应的计算方法，从而简化了编程。

■ MATLAB程序q806.m

```
clear, clf
wn=10; Ts=0.1                          % 设定参数 wn, Ts
for zeta=[0.1:0.3:1]               % 设定参数 zeta
    [num, den]=ord2(10, zeta);
    s=tf(num, den);                    % 建立连续系统 s
    sd=c2d(s, Ts);                     % 再生成采样系统 sd
    figure(1), impulse(s, 2), hold on      % 在图 1 中画 2 秒连续系统曲线
    figure(2), impulse(sd, 2), hold on     % 在图 2 中画 2 秒采样系统曲线
end                                                  % 标注语句略去
hold off
```

■ 程序运行结果

对连续和离散系统绘制的脉冲响应曲线分别见图 8.5-1 和图 8.5-2。

【例 8.7】 附加零点对二阶连续系统脉冲响应的影响

含有零点的二阶系统的传递函数为

$$H(s) = \frac{\omega_{n}^{2}(T_{m}s + 1)}{s^{2} + 2\zeta\omega_{n}s + \omega_{n}^{2}}$$

设其固有频率ω_n=1，阻尼系数ζ = 0.4，在 T_m = 0.5，1，2 时，分别画出其脉冲响应函数。将系统在条件 T_s = 0.1 下离散化，再做脉冲响应曲线。

解：用 ord2 函数在这里并不方便，故用 tf 函数建立此二阶连续系统 LTI 模型 s，用 c2d 函数转换为离散 LTI 模型 sd，再用 impulse 函数绘制脉冲响应曲线。不同的ζ 用 for 循环处理。

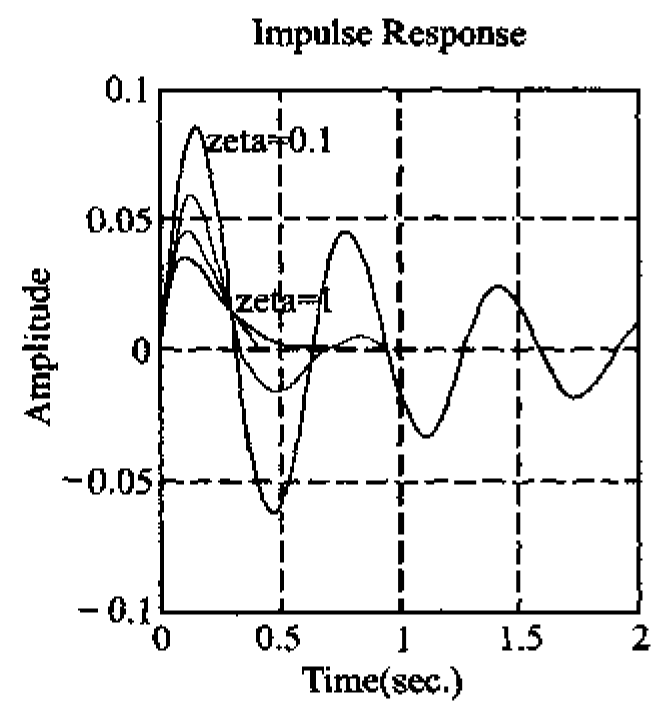


图 8.5-1 连续系统在不同ζ值下的脉冲响应曲线

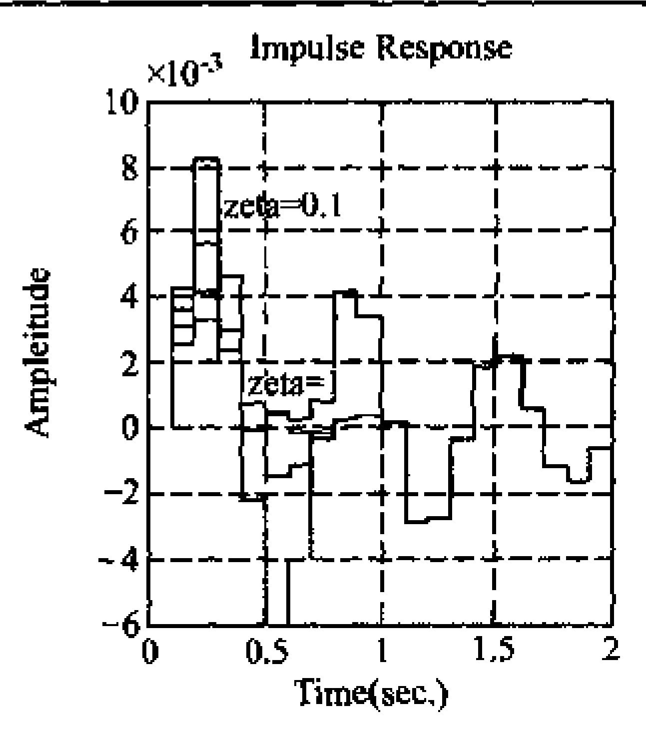


图 8.5-2 离散系统在不同ζ值下的脉冲响应曲线

■ **MATLAB程序q807.m**

```
clear, clf
wn=1; Ts=0.5; zeta=0.4                           % 设定参数 wn, Ts
for Tm=[0.5, 1, 2]                               % 设定参数 zeta 的三个值
    s=tf([Tm, 1]*wn^2, [1, 2*zeta*wn, wn^2]);           % 建立连续系统 s
   sd=c2d(s, Ts);                                % 用零阶保持器生成采样系统 sd
   figure(1), step(s), hold on           % 在图 1 中画连续系统曲线
   figure(2), step(sd), hold on          % 在图 2 中画采样系统曲线
end
hold off
```

■ **程序运行结果**

对连续系统和离散系统绘制的阶跃响应曲线分别见图 8.6-1 和图 8.6-2。

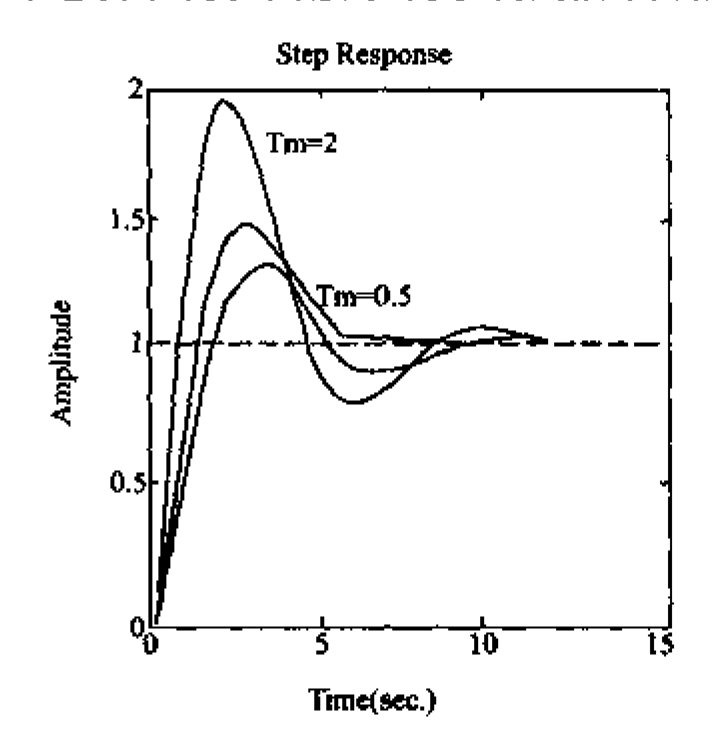


图 8.6-1 连续系统在不同ζ值下的阶跃响应曲线

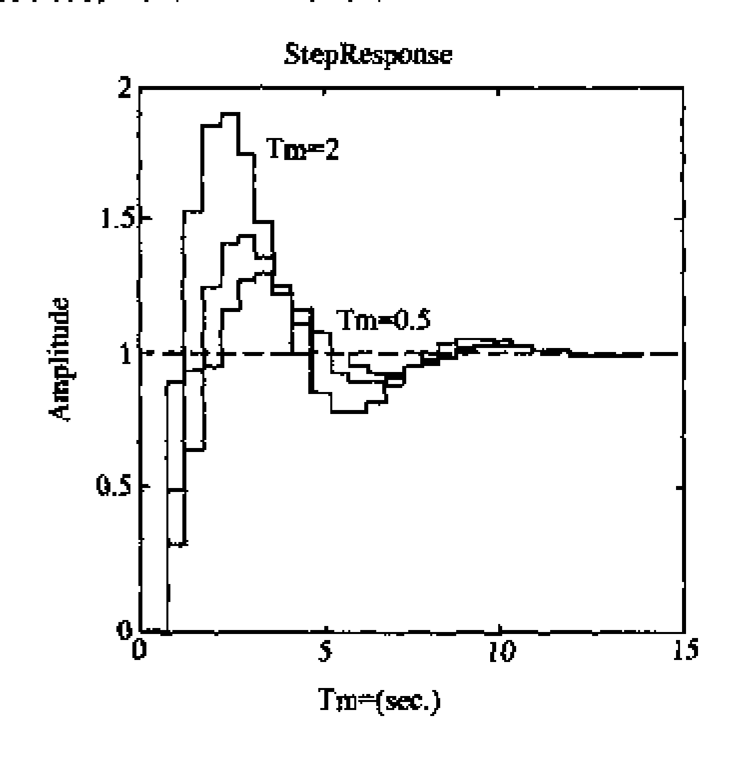


图 8.6-2 离散系统在不同ζ值下的阶跃响应曲线

可见，所加的零点越小，即时常数 T_m 越大，则阶跃过渡过程的超调加大，上升时间减小，使系统的跟踪速度加快。

【例 8.8】 附加极点对二阶连续系统脉冲响应的影响

含有附加实极点$1/T_p$的二阶系统的传递函数为

$$H(s)=\frac{\omega_n^{\ 2}}{(s^2+2\zeta\omega_n s+\omega_n^{\ 2})(T_p s+1)}$$

设其固有频率$\omega_n=1$，阻尼系数$\zeta=0.4$，设T_p= 0.5，1，2，分别画出其脉冲响应函数和极点分布。并进行讨论。

解：用 tf 函数建立此三阶连续系统 LTI 模型 s，再用 impulse 函数绘制脉冲响应曲线。用 pzmap 函数绘制零极点分布，不同的附加极点用 for 循环处理。

■ **MATLAB程序q808.m**

```
clear, clf
wn=1; zeta=0.4                                % 设定参数 wn,  zeta
for Tp=[0.5, 1, 2]                            % 设定参数 Tp
    den=conv([Tp, 1], [1, 2*zeta*wn, wn.^2]);
    s=tf(wn^2, den);                          % 建立连续系统 s
    figure(1), step(s), hold on               % 在图 1 中画阶跃响应曲线
     figure(2), pzmap(s), hold on             % 在图 2 中画零极点分布图
    w=1; P=covar(s, w)                        % 计算密度为 1 的白噪声通过系统后的输出方差
end
hold off
```

■ **程序运行结果**

对含不同的附加极点系统绘制的阶跃响应曲线分别见图 8.7-1 和图 8.7-2。

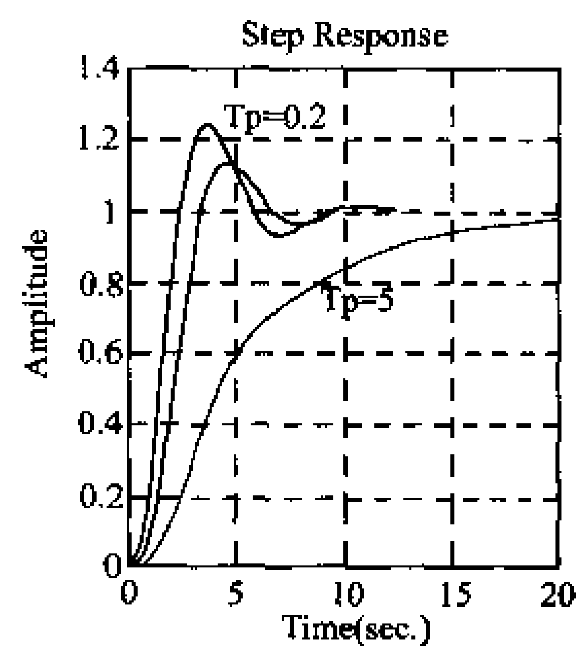


图 8.7-1　系统在不同附加极点下的阶跃响应曲线

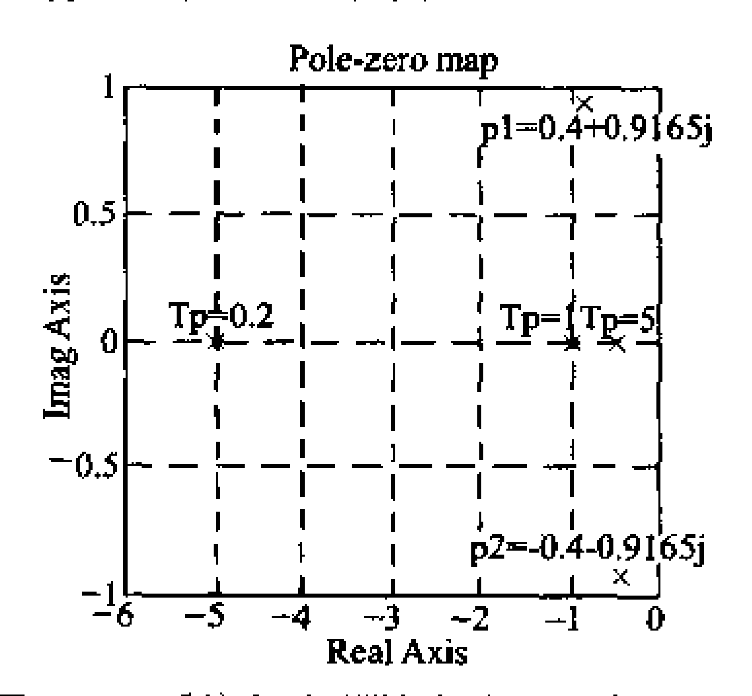


图 8.7-2　系统在不同附加极点下的零极点分布图

对应于T_p=0.2，1，2 的输出方差分别为P=0.6042，0.4018，0.1042。可见，附加的极点越小，即时常数T_p越大，则阶跃过渡过程的上升时间加大。这将使系统的跟踪速度减慢，同时对噪声的抑制能力增大。在T_p=5，即实极点模为 0.2，也就是复极点的 1/5 时，系统的阶跃过渡过程基本上由这个实极点所决定。而当T_p =0.2，即极点模为 5，也就是两个复极点的 5 倍时，系统的阶跃过渡过程基本上由两个复极点所决定。可以近似认为，系统的响应主要取决于虚部最小的极点。特别是当其他极点比它们大 5 倍以上时，这个或这对虚部最小的极点被称为主导极点。MATLAB 中的 esort 之所以按极点的虚部降秩排序，就是为了把主导极点排在前面。

【例 8.9】 系统的多种响应曲线

随机生成一个四阶 SISO 连续系统，求出它的状态方程，并分别画出它的（1）脉冲响应曲线，（2）初始条件为 $x_0 = [1, -1, 0, 2]$ 下的零输入响应曲线；（3）在正弦激励下的输出响应曲线；（4）系统的零极点分布图。将此系统离散化后，再分别画以上四种曲线。

解：■ 建模

这个例题要用到 initial 和 lsim 函数，它们也都同时适用于连续系统和离散系统。同时这里也采用了 rmodel 函数来随机生成一个稳定的四阶系统参数，所以做起来比较简单。编程时要注意的是：initial 函数只能用于状态空间模型。用 lsim 函数时，对连续系统必须给出时间数组 t，离散系统时间数组 t 的步长必须与采样周期 Ts 相同。

■ MATLAB程序q809.m

```
clear, clf
[a, b, c, d]=rmodel(4);             % 生成一个随机四阶状态空间模型
s1=ss(a, b, c, d);Ts=0.2;           % 命名它为 s1，设定采样周期
sd1=c2d(s1, Ts, 't');               % 用双线性变换转换为采样系统
t=0:Ts:25;                          % 设定时间数组
u=sin(0.5*t);                       % 设定输入
for i=1:2                           % 用循环方法求两个系统特性
   if i==1 s=s1;
   else s=sd1; end
figure(i)
   subplot(2, 2, 1), impulse(s, 5);grid          % 求脉冲响应
   subplot(2, 2, 2), lsim(s, u, t);              % 求输入作用下的输出响应
   subplot(2, 2, 3), x0=[1, -1, 0, 2];           % 求初始条件响应
   initial(s, x0, 5), grid
   subplot(2, 2, 4), pzmap(s), grid              % 求零极点分布图
end
```

■ 程序运行结果

连续系统和离散系统的这些响应曲线分别见图 8.8-1 和图 8.8-2。因为这是随机生成的系统，每次运行的结果不同，读者上机得出的可能将是完全不同的曲线。

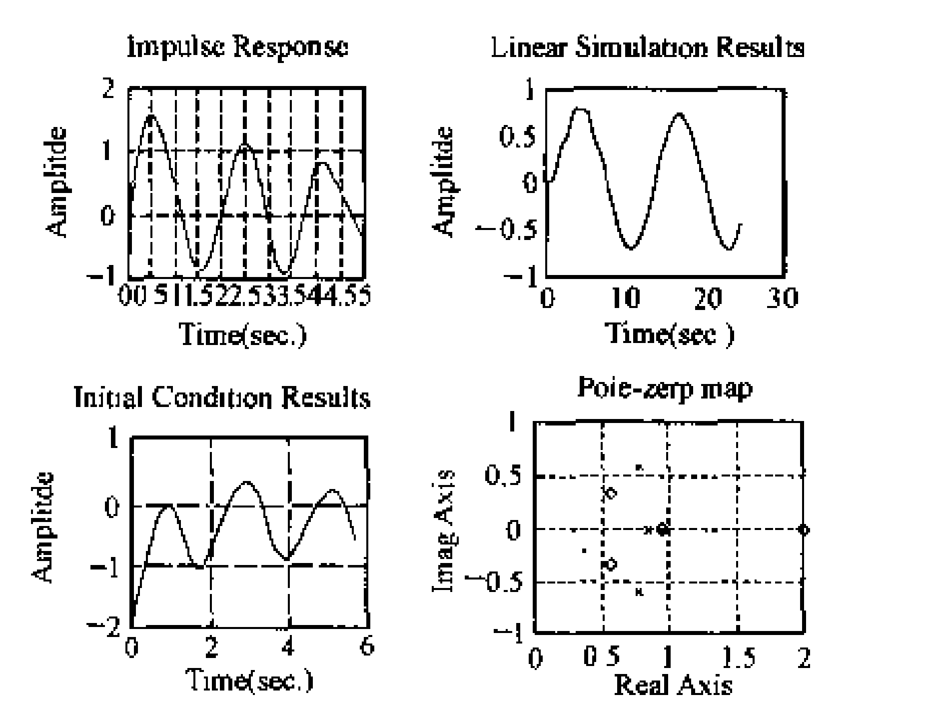


图 8.8-1　连续系统的多种响应曲线

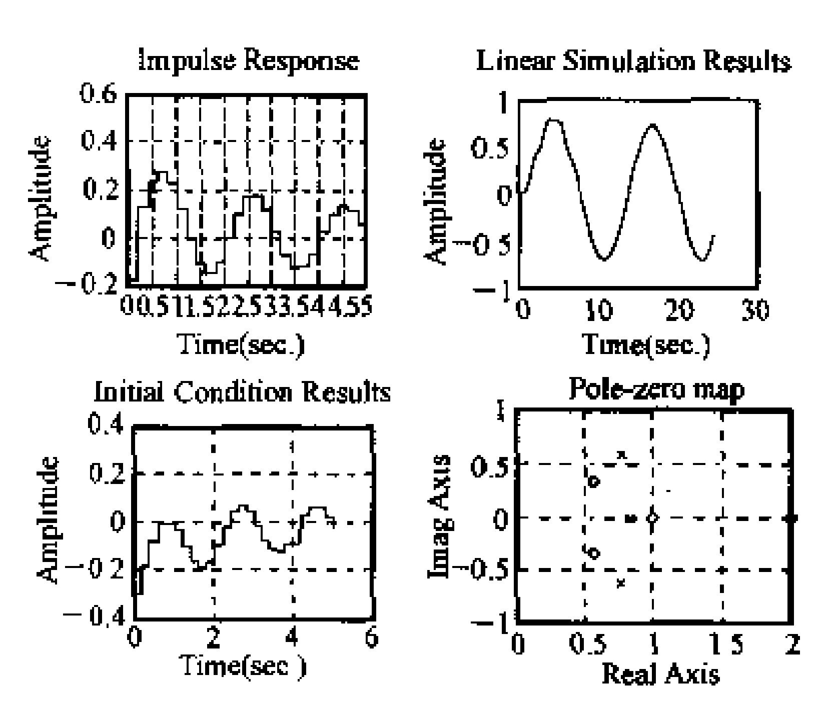


图 8.8-2　离散系统的多种响应曲线

注意这两张图的第四个子图，图上的叉号表示极点，圆圈表示零点。两张图的坐标也是不同的，连续系统的零极点画在 s 平面上，而离散系统的零极点画在 z 平面上。本例中系统在 s 平面上的根很接近虚轴，使其脉冲响应中有很强的振荡分量。注意其时间坐标，它与输入的正弦波不是一个频率。

【例 8.10】 带时延环节的系统分析

设系统的开环传递函数为

$$W(s)=\frac{K(T_m s+1)e^{-T_d s}}{s^2}$$

其中，$K=0.1[1/s^2]$，$T_m=5s$，$T_d=1s$，

将系统的环路闭合起来如图 8.9-1 所示。

分析它的极点变化情况，并求闭环系统的脉冲响应和阻尼系数。

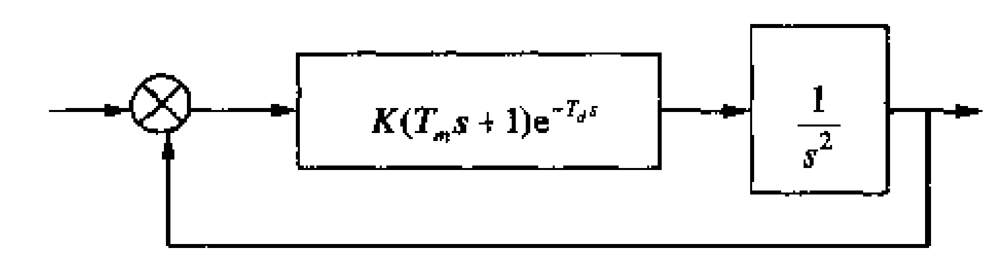


图 8.9-1　例 8.10 的图

解：首先建立系统在开环状态下的模型，这是一个带时延环节的二阶无静差系统，可写出：

s = tf(K*[Tm,1],[1,0,0],'Td',1)

然后用 feedback 命令把系统闭合起来（sb=feedback(s,1)）。此时 MATLAB 回答 feedback 命令不能用于带时延环节的系统。为解决这个问题，必须把时延环节近似为多项式，设用 pade 命令以四阶多项式来近似代替时延环节，改变开环系统的模型使之成为 spd = pade(s, 4)，然后再求闭环模型 sbpd = feedback(spd，1)。模型正确建立后，即可用有关命令观察系统开闭环极点的特性并求闭环系统的脉冲响应和阻尼系数。

■ MATLAB程序q810.m

```
K=0.1; Tm=5; Td=1;
s =  tf(K*[Tm, 1], [1, 0, 0], 'Td', Td)       % 建立系统在开环状态下的模型
spd=pade(s, 4)                                 % 以四阶多项式来近似代替延时环节的模型
sbpd=feedback(spd, 1)                          % 求闭环模型
damp(sbpd), pause                              % 求闭环模型的阻尼系数
subplot(2, 2, 1), pzmap(s), pause              % 画出原始开环模型 s 的零极点
subplot(2, 2, 2), pzmap(spd), pause            % 画出近似开环模型的零极点
subplot(2, 2, 3), pzmap(sbpd), pause           % 画出闭环模型的零极点
subplot(2, 2, 4), impulse(sbpd)                % 画出闭环环系统的脉冲响应
```

标注语句略去。

■ 程序运行结果

原始开环模型 s

```
Transfer function:
0.5 s  + 0.1
----------------
   s^2
```

```
Input delay: 1
```

pade 近似开环模型 spd

```
Transfer function:
0.5 s^5 - 9.9 s^4 + 88 s^3 - 402 s^2 + 756 s + 168
-----------------------------------------------------
  s^6 + 20 s^5 + 180 s^4 + 840 s^3 + 1680 s^2
```

pade 近似后的闭环模型 sbpd

```
Transfer function:
     0.5 s^5 - 9.9 s^4 + 88 s^3 - 402 s^2 + 756 s + 168
----------------------------------------------------------------------
s^6 + 20.5 s^5 + 170.1 s^4 + 928 s^3 + 1278 s^2 + 756 s + 168
```

闭环模型极点的固有频率和阻尼系数为

```
         Eigenvalue                Damping      Freq. (rad/s)
 -5.38e-001                         1.00e+000     5.38e-001
 -5.61e-001  +  3.81e-001i          8.27e-001     6.78e-001
 -5.61e-001     -3.81e-001i         8.27e-001     6.78e-001
 -3.37e+000  +  6.69e+000i          4.50e-001     7.49e+000
 -3.37e+000  -  6.69e+000i          4.50e-001     7.49e+000
 -1.21e+001                         1.00e+000     1.21e+001
```

可见，系统是稳定的，但阻尼系数偏小，虚部最大的优势根有一个实根和一个复根，复根的阻尼系数还不到 0.1。这种含双积分的二阶无差系统是位置制导系统的典型结构，要使它稳定且有良好性能是要下功夫的。

得出的图形见图 8.9-2，第一个子图说明原始系统在 s 平面原点有一个双重极点，而在 $1/T_m$ 处有一个实零点；第二个子图说明近似开环系统多了四个左半平面极点和四个右半平面零点，它们基本上是处在一个半径约为 $N\pi/2T_d$ 的圆上（N 为 pade 的阶数）。这是四阶 pade 近似造成的。第三个子图说明系统反馈以后的极点，第四个子图为闭环系统的脉冲响应。

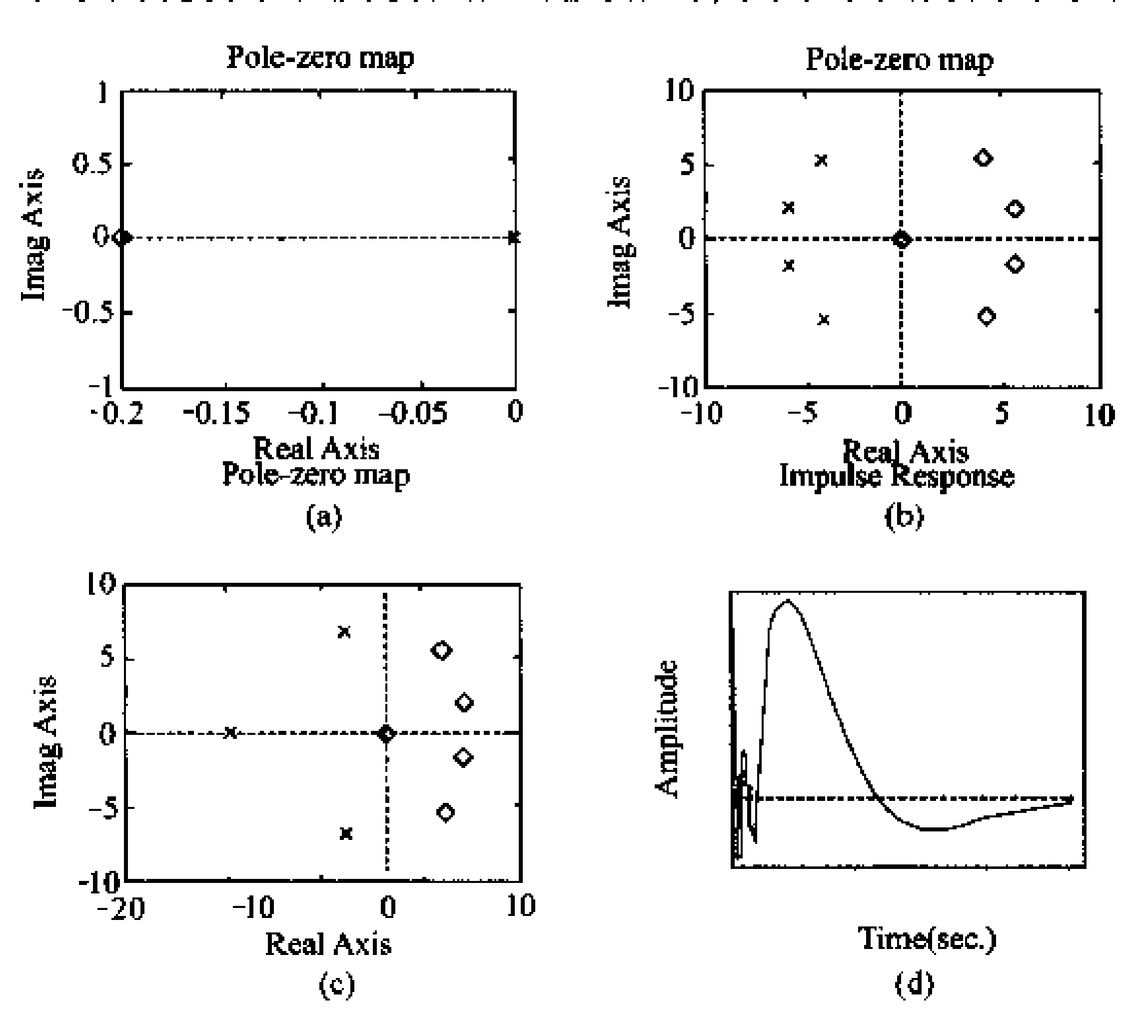


图 8.9-2　(a)原始系统零极点 (b)pade 近似系统零极点 (c)闭环系统零极点 (d) 闭环系统脉冲响应

【例 8.11】 连续和离散系统的根轨迹绘制

设系统的开环传递函数为

$$H(s)=\frac{1}{s^4+12s^3+30s^2+50s}$$

画出系统的根轨迹，并求出临界点（即根在虚轴上）的增益。

设 T_s= 0.5，将系统离散化后，做同样的工作。

解：■ 建模

先建立系统的 LTI 连续模型 s，然后用 rlocus(s)函数画它的根轨迹，再键入 rlocfind(s)函数，用鼠标选择根轨与虚轴的交点，即临界点。然后转成系统的 LTI 离散模型 sd。要注意连续系统和离散系统根平面之间的映射关系。s 平面的左半平面映射为 z 平面单位圆的内部。s 平面上的虚轴映射为 z 平面单位圆边界，s 平面上的原点映射为 z 平面上的点（1，0），而 s 平面上的无穷远点映射为 z 平面上的点（−1，0）。掌握这些要点就能够比较出两种情况根轨迹的对应关系。

■ MATLAB程序q811.m

```
clear, clf
disp('先分析连续系统'), pause
s=tf(1, [1, 12, 30, 50, 0])        % 建立连续系统开环模型 s
figure(1), rlocus(s)               % 连续系统 s 的根轨迹绘制
sgrid                              % 画出 s 平面网格线
rlocfind(s)                        % 用鼠标求临界点及 k 值
disp('以下分析离散系统'), pause
sd=c2d(s, 0.5, 't')                % 用双线性变换建立离散系统开环模型
figure(2), rlocus(sd)              % 离散系统 sd 的根轨迹绘制
zgrid                              % 画出 z 平面网格线
rlocfind(sd)                       % 用鼠标求临界点及 k 值
```

■ 程序运行结果

连续系统 s 的根轨迹见图 8.10-1，离散系统 sd 的根轨迹见图 8.10-2。

对连续系统用鼠标求临界点的根为

```
selected_point =   0.0046 + 1.9883i
```

k 值为

```
 ans =  103.5429
```

对离散系统用鼠标求临界点的根为

```
selected_point =   0.5806 + 0.7953i
```

k 值为

```
 ans =  103.0339
```

两个 k 值的微小差别可能有多种原因造成。首先是鼠标器的取值不可能很准确，其次是连续系统离散化以后的临界增益发生了变化，一般说来，连续系统经过采样以后再闭环，采样器的延时会使系统的稳定性下降。本题用的双线性变换对稳定性的影响比较小，若用零阶保持器，影响要大得多。

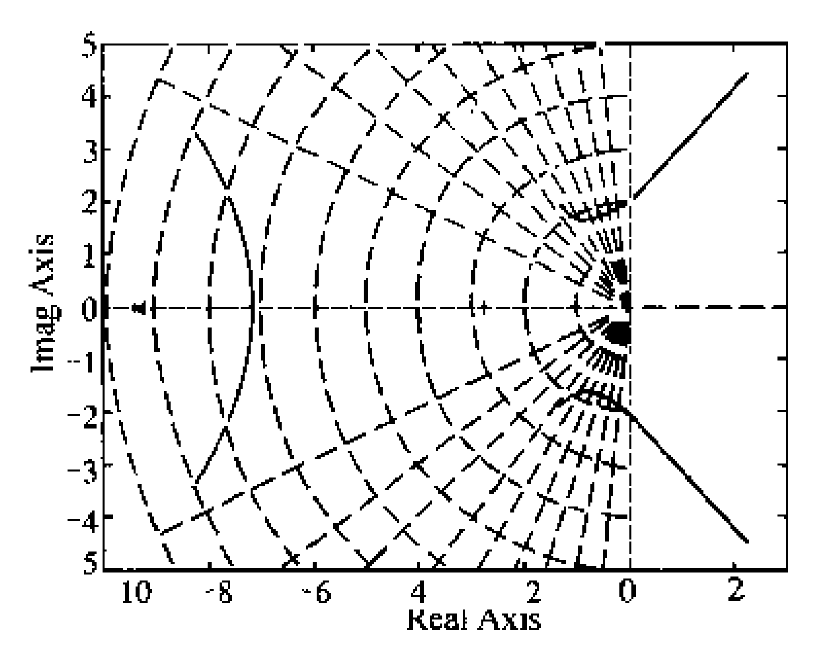


图 8.10-1　连续系统 s 的根轨迹

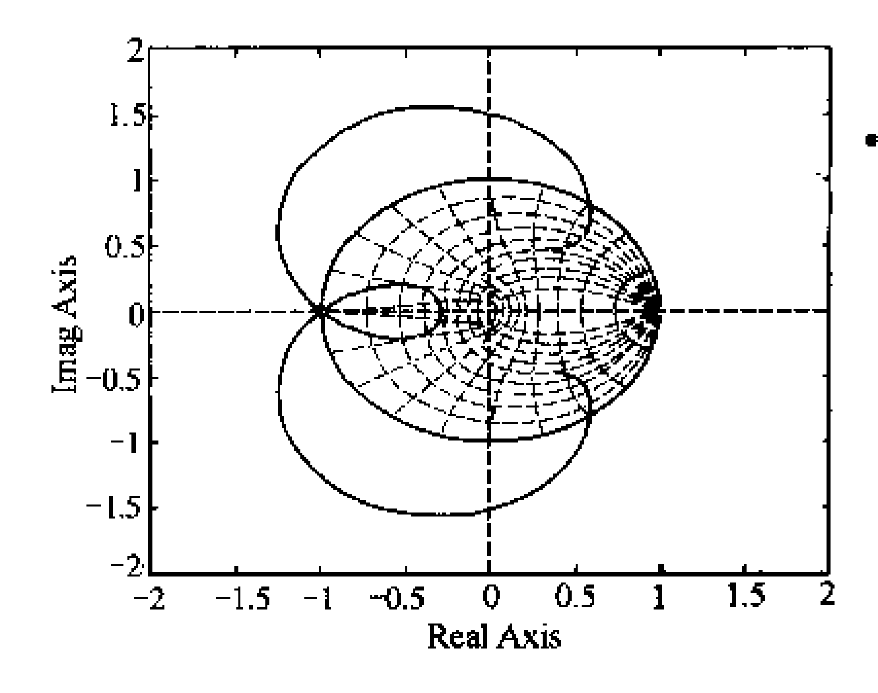


图 8.10-2　离散系统 sd 的根轨迹

【例 8.12】 带时延环节的系统根轨迹分析

系统的结构图同例 8.11，设开环传递函数为

$$W(s)=\frac{K(T_m s+1)e^{-T_d S}}{s^2}$$

其中，K=0.1[1/s²]，T_m=5s，时延 T_d=1s，

要求绘制其根轨迹，找到阻尼系数最大的主导共轭极点并确定此时系统的开环增益 K，并绘出其脉冲响应。

解：在用根轨迹解决问题的时候，通常要有一个交互的过程，因此不能指望靠一个编好的程序执行到底，而要在命令窗中，根据显示的结果，不断键入新的命令才行。

首先建立系统在开环状态下的模型，它是一个带时延环节的二阶无静差系统，因为根轨迹函数 rlocus 不能用于带时延环节的系统，必须把时延环节近似为多项式，即用 pade 命令。若以六阶多项式来近似代替时延环节，则开环系统的近似多项式模型成为 spd = pade(s，6)，然后调用根轨迹函数 rlocus 绘制根轨迹，为了找到阻尼系数最大的主导极点，要用 rlocfind 函数，用人机交互确定 K 后，再构成闭环系统并求其脉冲响应。

■ MATLAB程序q812.m

```
K=0.1;Tm=5;Td=1;
s =  tf(K*[Tm, 1], [1, 0, 0], 'Td', Td)          % 建立系统在开环状态下的模型
spd=pade(s, 6)                             % 以四阶多项式来近似代替时延环节的模型
figure(1), rlocus(spd)                     % 绘制根轨迹
```

■ 程序运行过程

```
Transfer function:
0.5 s + 0.1
-----------
    s^2
Input delay: 1
Transfer function:
```

```
0.5 s^7 - 20.9 s^6 + 415.8 s^5 - 4956 s^4 + 36792 s^3 - 158760 s^2  + 299376 s + 66528
----------------------------------------------------------------------------------------
 s^8 + 42 s^7 + 840 s^6 + 10080 s^5 + 75600 s^4 + 332640 s^3 + 665280 s^2
```

同时得到图 8.11-1 所示的根轨迹。

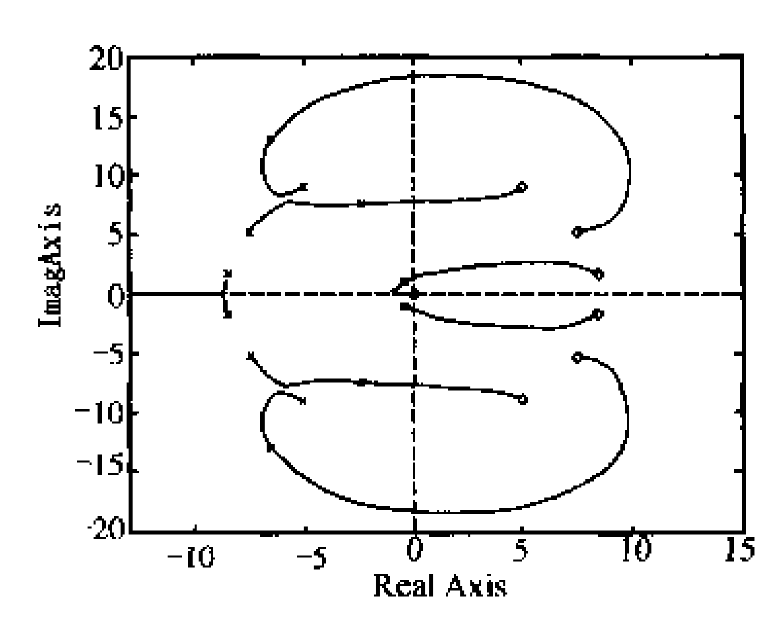


图 8.11-1　系统根轨迹及初选的闭环极点

系统开环有八个极点，共有八条根轨，我们关心的主导极点是离原点最近的复数极点，在此图上看不太清楚，可以键入

```
rlocfind(spd)
```

初步用鼠标在离原点最近的根轨上选一点。这时命令窗中出现

```
Select a point in the graphics window
selected_point =
  -0.4839 - 1.0526i
ans =
      1.5177
```

表示选定的复数极点及其对应的 k 值。同时在根轨图上出现八个叉号，显示在此 k 值下系统闭环极点的分布情况。可以看出这些极点都在左半平面，说明系统是稳定的，因此，可以专注于这两个小极点附近去仔细微调。为此，把根轨坐标放大，键入

```
axis([-3, 3, -3, 3])
```

得到图 8.11-2，这个根轨太粗，要仔细地画一下。为此把 K 细分为 K=[0:0.1:1.5]，重画根轨。

键入

```
rlocus(spd, 0:0.1:1.5);
```

得到图 8.11-3，再改变比例尺并键入

```
axis([-2, 2, -2, 2])
```

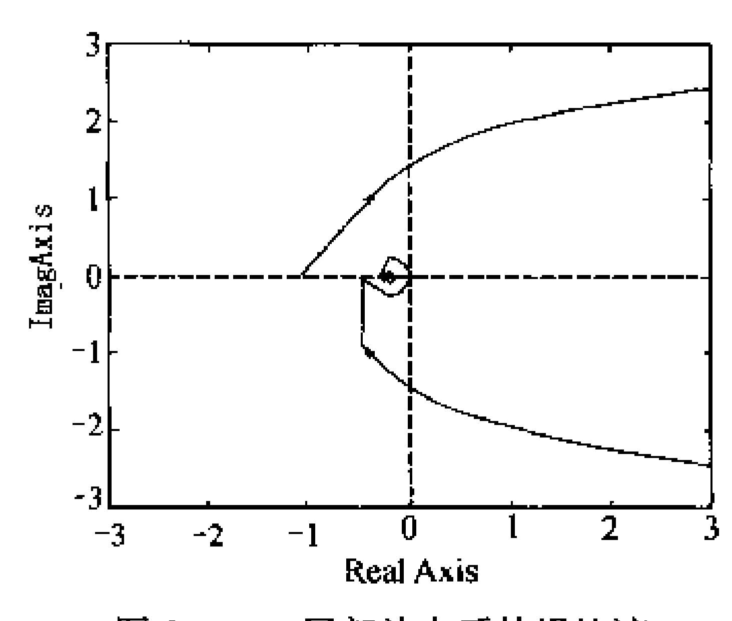


图 8.11-2　局部放大后的根轨迹

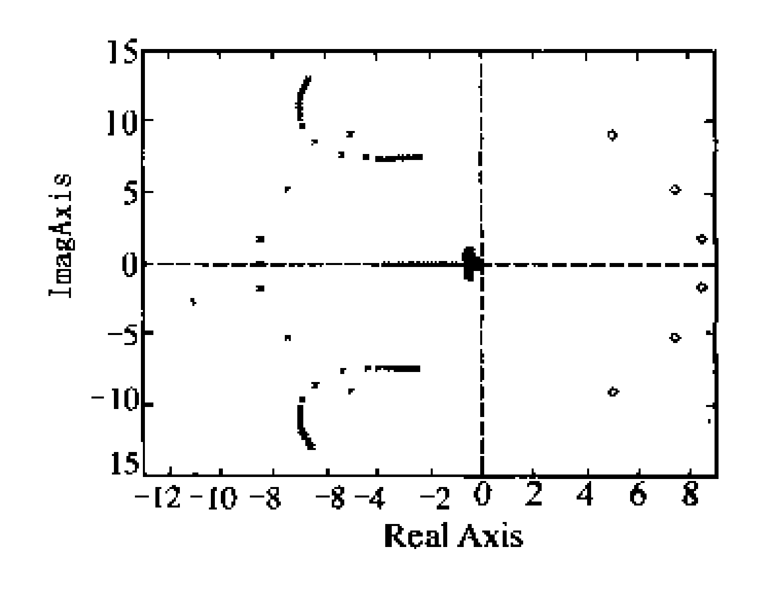


图 8.11-3　*K* 加密后的根轨迹

为了便于选择阻尼系数最大的点，要加上网格函数。再键入

```
sgrid
```

得到图 8.11-4 后，再用鼠标选极点。键入

```
rlocfind(spd)
```

图形屏幕出现十字线，选择根轨与阻尼系数最大的网线相切的点，按鼠标左键。命令窗出现

```
selected_point = -0.5115 + 0.2807i
ans =0.9736
```

就是说，最佳的增益将在 k = 0.9736 时。注意这个 k 不是题中的 K，而是指在系统 spd 的系统函数上应该乘的增益。为了求出取这个 k 值时的全部闭环极点，有多种方法，这里用 **rlocus** 的另一格式，键入

```
r=rlocus(spd, 0.9736)
```

得到

```
r = -21.5633
  -6.8325+12.2057i
  -6.8325-12.2057i
  -2.7817+7.4437i
  -2.7817-7.4437i
  -0.6979
  -0.4986+0.3158i
  -0.4986-0.3158i
```

再键入不带左端变元的格式

```
rlocus(spd, 0.9736)
```

就得到闭环根的最后分布图 8.11-5。按这个 k 值闭合系统，并求其脉冲响应，可再键入

```
sbpd=feedback(spd, 1);
impulse(sbpd)
```

所得脉冲响应大体上与图 8.9-2 相似，因为所乘的 k 很接近于 1。也可用 **damp(sbpd)** 语句直接得到所有根的阻尼系数和固有频率。

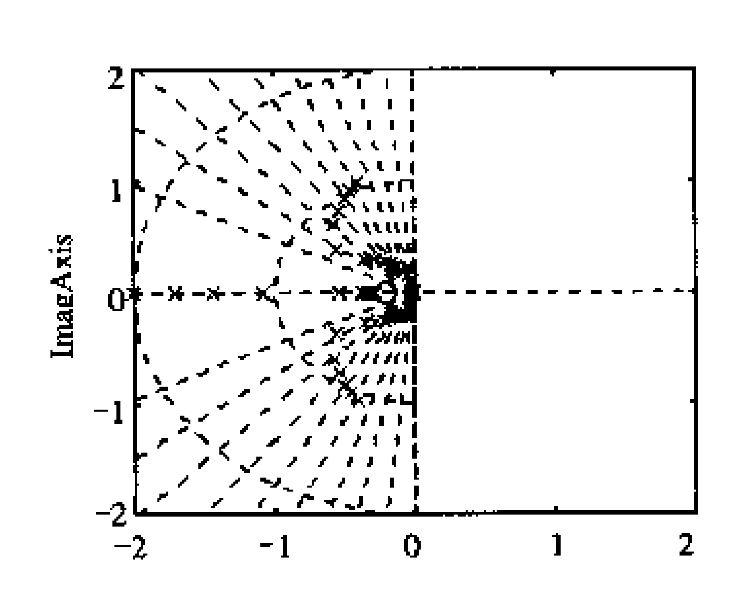


图 8.11-4　用加密的 k 并放大后的根轨迹

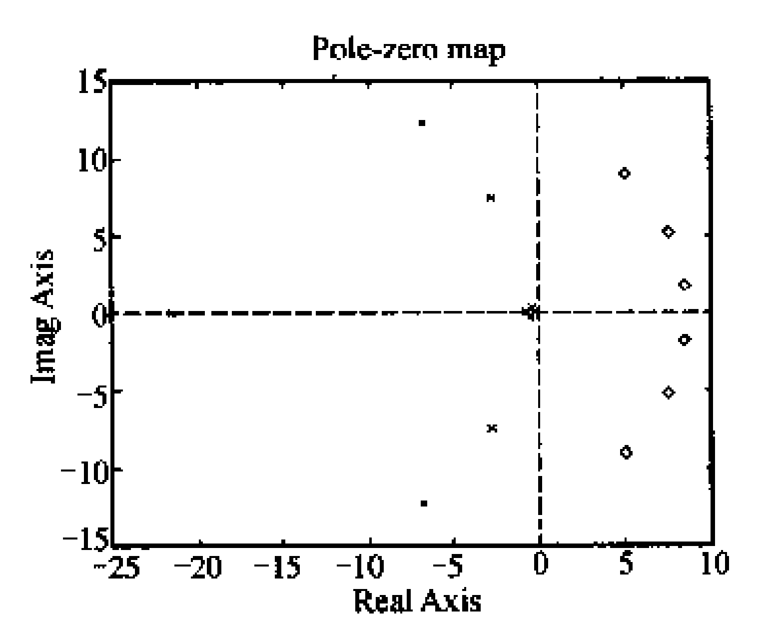


图 8.11-5　k 取 0.9736 的根分布

8.3　系统的频域分析函数

控制工具箱中的频域分析函数参见表 8.7。

这些函数中凡是以系统名称 sys 作为输入变元的，都同时适用于连续系统和离散系统。而且也适用于多输入多输出系统，因为这些特征都已包含在系统名称中。

表 8.7　控制工具箱中的频域分析函数

参数名称和典型调用格式	功　能
bode(sys)，[mag，phase，w]=bode(sys)	bode 图绘制
fres=evalfr(sys，f)	计算系统单个复频率点 f 的频率响应
H=freqresp(sys，w)	计算系统在给定实频率区间的频率响应
[Gm，Pm，wcg，wcp]=margin(sys)　margin(sys)	计算系统的增益和相位裕度
ngrid	Nichols 网格图绘制
nichols(sys)	Nichols 图绘制
nyquist(sys)	Nyquist 图绘制
sigma(sys)	系统奇异值 bode 图绘制

➢ bode(sys)　用来计算系统的对数频率响应，画出 bode（波德）图，但不返回数据（不管 sys 是连续系统还是离散系统）。如果是多输入多输出系统。画出的 bode 图将自动分成相应的子图。

有左端输出变量 [mag, phase, w] = bode(sys) 时，它计算并返回系统对数频率响应的振幅、相位和对应的频率数据，但不返回图形。

bode(sys, w) 中变元 w 可以规定绘图的频率范围或频点。

bode(sys1, sys2, …, sysN, w) 可以在一张图上画出多个系统的 bode 图。

bode(sys1, 'PlotStyle1', sys2, 'PlotStyle1'…, w) 可以规定多个系统的 bode 图的外观。

➢ H = evalfr(sys, f)　计算系统 sys 单个复频率点 f 的频率响应。其公式为

$$H(f)=D+(C\,(fI-A)^{-1}B$$

f 应以复数标量给出。

➢ H = freqresp(sys, w)　计算系统在给定实频率区间 w 的频率响应 H，其中 w 为实数数组，H 则为复数数组，对多输入多输出系统，H 是三维复数矩阵。

➢ [Gm, Pm, wcg, wcp] = margin(sys)　计算系统的增益裕度 Gm、相位裕度 Pm 和相应的穿越频率 wcg，wcp。

➢ [Gm, Pm, wcg, wcp] = margin(mag, phase, w)　表示输入变元也可以是 bode 图的输出数据。

margin(sys) 在无左端输出变量时，给出 bode 图及穿越频率处的标志，在图上给出数据。

➢ ngrid　用于 nichols 网格图绘制，如图 8.12 所示。

➢ nichols(sys)　用于 nichols 图绘制。

➢ nyquist(sys)　用于 nyquist 图绘制。

➢ sigma(sys)　用于系统奇异值 bode 图绘制。

这几种频率特性图的绘制函数与 bode 图相仿，bode 的调用格式也都适用于这几个函数。nichols 图和 nyquist 图在一般的自动控制课程中都要介绍，奇异值 bode 图是鲁棒控制理论

中用到的概念。在最简单的情况下，它就是 bode 图。

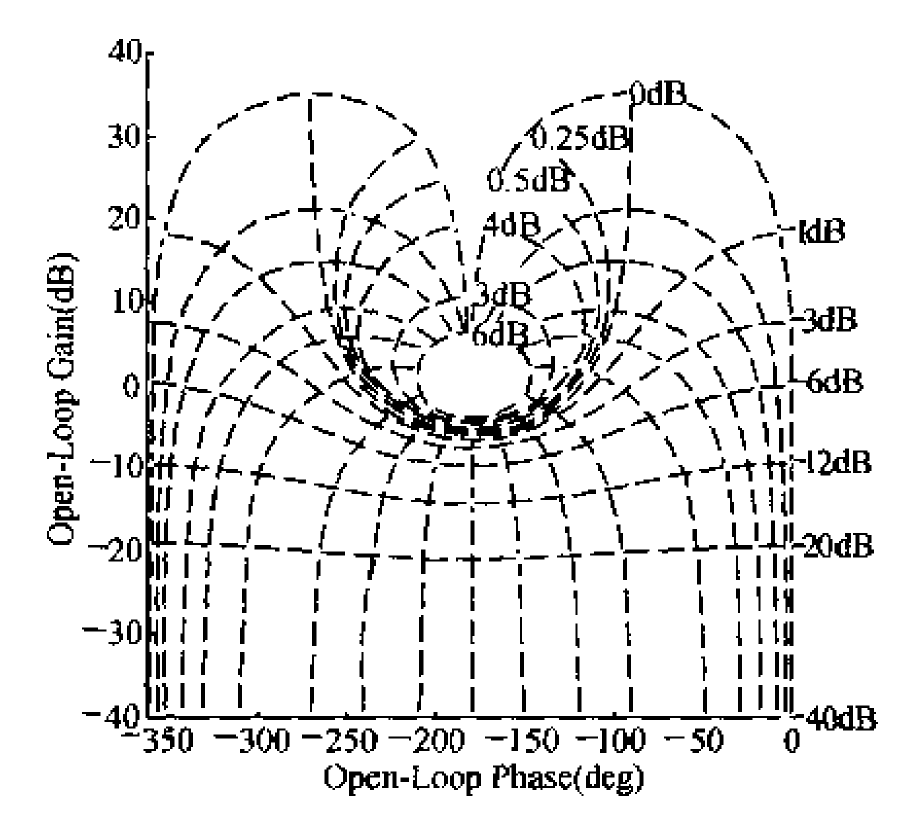


图 8.12 用 ngrid 命令绘制的 nichols 网格图

【例 8.13】 阻尼系数对二阶系统频率响应的影响

二阶系统的传递函数为

$$H(s)=\frac{1}{s^2+2\zeta\omega_n s+\omega_n{}^2}$$

设其固有频率$\omega_n=10$，阻尼系数ζ=[0.1，0.3，0.7，1]，分别画出其 bode（波德）图。将系统在条件 T_s=0.1 下离散化，并做同样的工作。

解：■ 建模

先用 ord2 函数建立二阶连续系统 LTI 模型 s，用 c2d 函数转换为离散 LTI 模型 sd，再用 bode 函数绘制对数频率特性曲线。不同的ζ 用 for 循环处理。注意 bode(sys) 函数对连续系统和离散系统是公用的，它会根据 sys 的不同属性自动选择相应的计算方法，从而简化了编程。

■ MATLAB程序q813.m

```
clear, clf, wn=10;
for zeta=[0.1:0.3:1]                 % 设定不同的 zeta
   [n, d]=ord2(wn, zeta);
   s1=tf(n*wn^2, d);                 % 生成二阶连续系统
   sd1=c2d(s1, 0.1);                 % 转成二阶采样系统，采样周期 0.1 秒
   figure(1), bode(s1), hold on      % 画波德图
   figure(2), bode(sd1), hold on     % 画波德图，注意离散系统用同样命令
end
hold off
```

■ 程序运行结果

对连续系统和离散系统绘制的波德图分别见图 8.13-1 和图 8.13-2。

从图中可以看出，二阶连续系统阻尼系数ζ 很小时，其幅频特性在转折频率处出现谐振峰，相频特性在这个频率附近迅速下降。随着ζ 的增大，幅频特性的峰值减小，在$\zeta\geqslant0.7$后，幅频特性单调下降。相频特性的下降也趋于平缓。

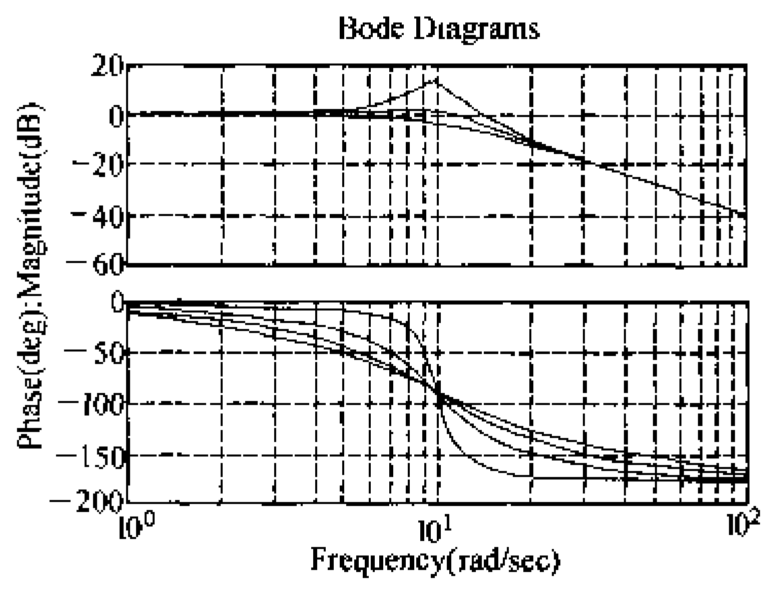
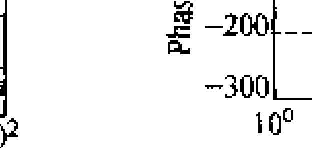


图 8.13-1　连续系统在不同 ζ 值下的波德图



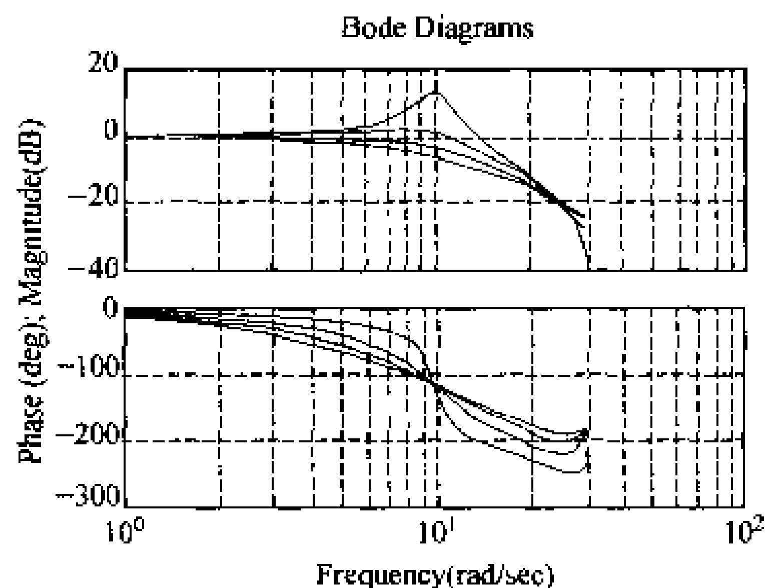


图 8.13-2　离散系统在不同 ζ 值下的波德图

离散系统的频率响应在低频区和连续系统基本相同，在高频区有一个最高极限频率，这个频率是采样频率的一半，在本例中采样周期 T_s= 0.1，故采样频率为 $2\pi/T_s$ = 62.8 [1/s]。图上的极限频率为 31.4 [1/s]。这体现了采样定理的规律，即经过采样以后的信号，只能保留半采样频率以下的频谱。不仅如此，从图中也可以看出，在靠近二分之一采样频率处的频率特性，也和连续系统差别较大，这是由于频率泄漏的效应，采样过程可以把高于半采样频率的一部分频谱，折叠相加到关于半采样频率点对称的频带上。造成了这部分频谱的畸变。所以实际设计采样开关时通常要把采样频率选为系统工作频率（注意是 f 而不是 ω）的 5~10 倍。

【例 8.14】 高阶系统的开闭环频率响应

设系统的传递函数为

$$H(s)=\frac{200(s+6)}{s(s+1)(s+10)^2}$$

画出其波德图。将系统在条件 T_s= 0.1 下离散化，并做同样的工作。然后把两种系统分别用单位负反馈构成闭合环路，画出其波德图与开环进行比较，并判断其稳定性。

解：■ 建模

先用zpk函数建立连续系统LTI模型s，用c2d函数转换为离散LTI模型sd，再用feedback函数得到两种系统的闭环传递函数，最后用bode函数绘制对数频率特性曲线。用damp(sys)函数判稳。注意damp(sys)函数对连续系统和离散系统也是公用的，它会根据sys的不同属性自动选择相应的计算方法，但判稳准则不同。对连续系统，阻尼系数大于零为稳定；对离散系统，根的模小于1为稳定。

■ MATLAB程序q814.m

```
clear,
Ts=0.1
s=zpk(-6, [0, -1, -10, -10], 200)          % 生成四阶连续系统 s
sd=c2d(s, Ts)                              % 变换为四阶离散系统 sd
sb=feedback(s, 1)                          % 把连续系统闭合，生成闭环系统 sb
sbd=feedback(sd, 1)                        % 把离散系统闭合，生成闭环系统 sbd
figure(1), bode(s, '--', sb, '.-')         % 在图 1 中绘出连续系统的开闭环频率特性
figure(2), bode(sd, '—', sbd, '.-')        % 在图 2 中绘出离散系统的开闭环频率特性
damp(sb)                                   % 闭环 sb 的根的固有频率和阻尼系数
damp(sbd)                                  % 判断闭环 sbd 的根的固有频率和阻尼系数
```

```
[Gm, Pm, wcg, wcp]=margin(s)          % 判断闭环 sb 的稳定裕度
[Gmd, Pmd, wcgd, wcpd]=margin(sd)     % 判断闭环 sb 的稳定裕度
```

■ **程序运行结果**

原始四阶连续系统s

```
Zero/pole/gain:
   200 (s+6)
-------------------------
s (s+1) (s+10)^2
```

对应的离散系统sd

```
Zero/pole/gain:
0.023516 (z+2.649) (z-0.5488) (z+0.1787)
-----------------------------------------------------
     (z-1) (z-0.9048) (z-0.3679)^2
Sampling time: 0.1
```

连续系统闭合后生成闭环系统sb

```
Zero/pole/gain:
                200 (s+6)
-----------------------------------------------------
(s+13) (s+7.513) (s^2 + 0.491s + 12.29)
```

离散系统闭合后生成的闭环系统sbd

```
Zero/pole/gain:
 0.023516 (z+2.649) (z-0.5488) (z+0.1787)
-----------------------------------------------------
(z-0.4869) (z-0.2366) (z^2 - 1.894z + 1.01)
Sampling time: 0.1
```

连续闭环sb的根的固有频率和阻尼系数

```
         Eigenvalue                Damping        Freq. (rad/s)
 -2.46e-001  +  3.50e+000i        7.00e-002          3.51e+000
 -2.46e-001  -  3.50e+000i        7.00e-002          3.51e+000
 -7.51e      +  000               1.00e+000          7.51e+000
 -1.30e      +  001               1.00e+000          1.30e+001
```

离散闭环sbd的根的固有频率和阻尼系数

```
         Eigenvalue            Magnitude     Equiv. Damping     Equiv. Freq. (rad/s)
  9.47e-001  +  3.37e-001i     1.01e+000      -1.47e-002          3.42e+000
  9.47e-001  -  3.37e-001i     1.01e+000      -1.47e-002          3.42e+000
  4.87e-001                    4.87e-001       1.00e+000          7.20e+000
  2.37e-001                    2.37e-001       1.00e+000          1.44e+001
```

闭环sb的稳定裕度

```
Gm = 1.7762,      wcg = 4.6560      Pm = 8.0194     wcp =     3.4467
```

离散闭环 sbd 的稳定裕度

```
Warning: Closed loop is unstable.
Gmd =  0,      Pmd = 0,       wcgd =  NaN,   wcpd =  NaN
```

对四阶连续系统和离散系统绘制的波德图分别见图 8.14-1 和图 8.14-2。

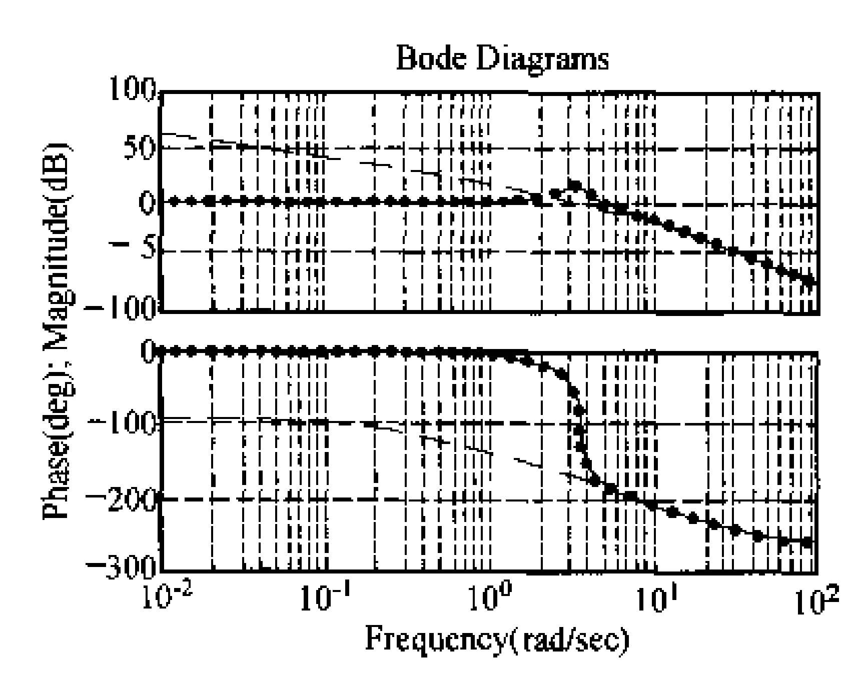


图 8.14-1　例 8.14 四阶连续系统的波德图

图 8.14-2　对应的四阶离散系统的波德图

由结果可知本题中的连续系统是稳定的，但稳定裕度很小，振幅稳定裕度 Gm 只有 1.7762dB，相位稳定裕度 Pm 为 8.0194° 。而对应的离散系统则是不稳定的。控制工具箱函数有很多都可以用来判别系统的稳定性，稳定裕度函数 margin 还会直接告诉用户系统稳定与否，因此实际上没有必要再去用画 nyquist 频率特性的方法来判别稳定性。仅仅在没有适当的计算工具来求复杂系统的闭环根时，这种古典的方法才有价值。

在分析系统的开环特性和闭环特性的关系时，可以注意两点，第一点，在开环的幅频特性远远大于 0dB（例如+20dB 以上）的低频区，闭环幅频特性为 0dB，相频特性也近似为零，说明在这个频段，反馈信号很强，这时闭环特性近似于反馈回路特性的逆；第二点，在开环的幅频特性远远小于 0 dB（例如−20dB 以下）的高频区，闭环与开环幅频特性和相频特性相重合，等于不起作用。读者可从开闭环传递函数变换关系中得出这个具有普遍意义的结论。

【例 8.15】 奈奎斯特曲线及判稳

设系统的传递函数为

$$H(s)=\frac{50}{(s-1.2)(s+1)(s+6)}$$

这是一个开环不稳定的系统。画出其 Nyquist 曲线，判别其闭环稳定性，并用 MATLAB 的其他函数加以检验。在此系统上加一个零点$(s+0.5)$后，再做同样的工作，把两种情况进行比较并讨论。

解：■ 建模

先用 zpk 函数建立系统的 LTI 模型 s1，调用 nyquist 函数画出其 nyquist 曲线，再用 feedback 函数得到它的闭环传递函数，用 margin 函数绘制出简略的对数频率特性曲线。用求闭环脉冲响应的方法再检验判稳的正确性。对系统 2 做同样的工作。

■ MATLAB程序q815.m

```
clear,
s1=zpk([ ], [-6, -1, 1.2], 50 );                % 生成连续系统 s1
```

```
figure(1)
subplot(2, 2, 1), nyquist(s1)     , grid          % 画 nyquist 图
sb1=feedback(s1, 1)
subplot(2, 2, 2), impulse(s1), grid               % 画开环脉冲响应图
subplot(2, 2, 3), margin(s1) , grid               % 画 nichols 图
subplot(2, 2, 4), impulse(sb1), grid              % 画闭环脉冲响应图
s2=zpk([-.5], [-6, -1, 1.2], 50 );                % 生成加微分的连续系统 s2
figure(2)
subplot(2, 2, 1), nyquist(s2), grid               % 画 nyquist 图
sb2=feedback(s2, 1)
subplot(2, 2, 2), impulse(s2), grid               % 画开环脉冲响应图
subplot(2, 2, 3), margin(s2),  grid               % 画 nichols 图
subplot(2, 2, 4), impulse(sb2), grid              % 画闭环脉冲响应图
```

■ **程序运行结果**

s1闭环后构成的系统sb1零极增益模型为

```
Zero/pole/gain:
               50
----------------------------------------
(s+7.013) (s^2 - 1.213s + 6.103)
```

s2闭环后构成的系统sb2零极增益模型为

```
Zero/pole/gain:
          50 (s+0.5)
----------------------------------------
(s+0.3914) (s^2 + 5.409s + 45.48)
```

得到的图形分别见图 8.15-1 和图 8.15-2。

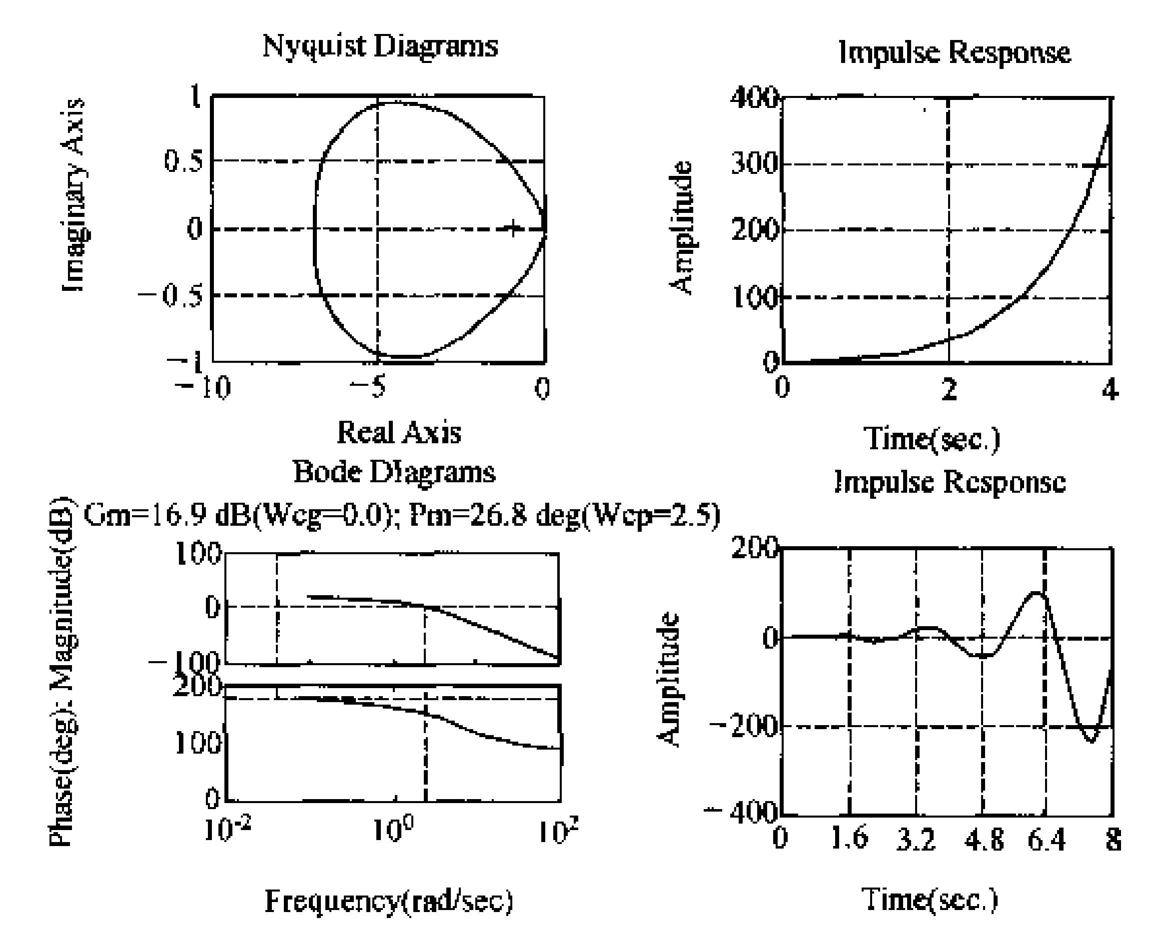


图 8.15-1　例 8.15 系统 1 的图形

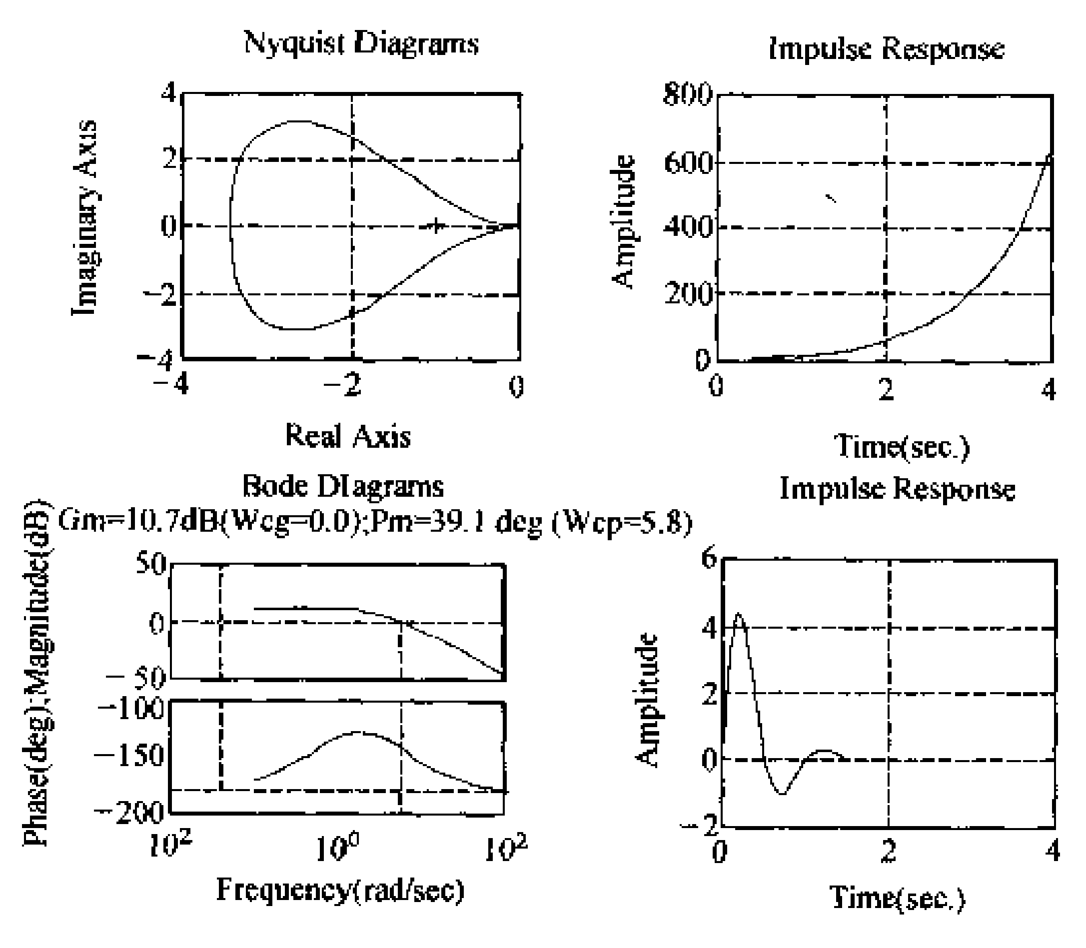


图 8.15-2　例 8.15 系统 2 的图形

从两个闭环模型的分母，已经清楚地看出，系统 1 是不稳定的，而系统 2 是稳定的。两个图的第二子图的脉冲响应，说明系统开环是不稳定的。从第四子图上，通过闭环脉冲响应，也可以得出同样的结论。而从奈奎斯特曲线上分析，就要费些功夫了。因为系统开环有一个右半平面极点，奈奎斯特曲线必须以反时针绕（−1，0）点转一圈，系统才是稳定的。系统 1 的奈奎斯特曲线是顺时针方向，因此是不稳定的；系统 2 的奈奎斯特曲线是反时针方向，因此是稳定的。两个图的第二子图为脉冲响应，说明两系统开环都是不稳定的。

这里还可以看出，没有输出变元的 margin 函数，不能够直接给出稳定的判据。它可以给出简略的波德图，并且在上面标注出振幅和相位裕度的值。但对于不稳定的系统它也给出这些值。很容易引起错觉。因此，如果没有用别的方法来判稳，还是应该用带有左端变元的 margin 函数来判稳，它会明确地指出系统是否稳定。

8.4　系统的状态空间分析函数

状态空间分析比其他方法之所以复杂，一是因为用矩阵进行运算和求解；二是因为它的非惟一性，即对同一个系统，通过相似变换，可以有无数种 ***A***，***B***，***C***，***D*** 组合来描述。MATLAB 控制工具箱提供的状态空间分析函数参见表 8.8。

● 第一类函数是关于系统可观性、可控性判别的，主要介绍以下几种。

（1）可控性矩阵

➢ Co = ctrb(sys)　或 Co = ctrb(A, B), …

其中 $\boldsymbol{Co} = [\boldsymbol{B}, \boldsymbol{AB}, \boldsymbol{A}^2\boldsymbol{B}, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{B}]$

若 rank(Co) = n，则系统可控。

（2）可观性矩阵

➢ Ob = obsv(sys)，Ob = obsv(A, C),…

其中
$$Ob = \begin{bmatrix} C \\ CA \\ \vdots \\ C^{n-1}A \end{bmatrix}$$

若 rank(Ob) = n，则系统可观。

表 8.8　控制工具箱中的状态空间分析函数

	函数和典型输入变元	功　　能
系统可观性、可控性	ctrb(sys)，ctrb(A，B)	计算系统的可控性矩阵
	obsv(sys)，obsv(A，B)	计算系统的可观性矩阵
	ctrbf(A，B，C)	可控阶梯形（可控与不可控）分解
	odsvf(A，B，C)	可观阶梯形（可控与不可观）分解
	gram(sys，'c')，	计算系统的可控或可观 Gramian 矩阵
相似变换和状态空间实现	ss2ss(sys，T)，	相似变换
	canon(sys，'type')	状态空间的规范实现
	ssbal(sys)	状态空间的均衡实现
	balreal(sys)	基于 Gramian 矩阵的状态空间均衡实现
	minreal(sys)	状态空间的最小实现
	modred(sys，elim)	状态空间的模型降阶

（3）Gramian 矩阵

系统可观性、可控性的另一判据，可用于时变系统。

可控 Gramian 矩阵，$Wc = \int_0^{\infty} e^{At}BB^{T}e^{A^{T}t}dt$

➢ Wc = gram(sys，'c')　　它的满秩(rank(Wc) = n)与系统的可控等价。

可观 Gramian 矩阵，$Wo = \int_0^{\infty} e^{A^{T}t}C^{T}Ce^{At}dt$

➢ Wo = gram(sys，'o')　　它的满秩 rank(Wo) = n 与系统的可观等价。

调用此函数时，要求系统必须是稳定的。

● 第二类函数是关于系统相似变换的。包括以下几种。

（1）通用相似变换函数

➢ sysT = ss2ss(sys，T)　　通过非奇异变换矩阵 T，把状态变量由 x 变为 xT。即 xT=Tx。则变换后的系统 sysT 状态方程系数矩阵为 AT，BT，CT，DT，即有 sysT=ss(AT，BT，CT，DT)，其中

$$AT = TAT^{-1},\quad BT = TB,\quad CT = CT^{-1},\quad DT = D$$

（2）变为规范形式的函数

➢ csys = canon(sys，'type')　　用来把系统 sys 变为规范形式 csys。type 用来选择规范的类型，有两种可选规范形式：约当矩阵（model）形式和伴随矩阵（companion）形式。要得到变换矩阵 T，可用以下格式

[csys，T] = canon(sys，'type')

（3）把系统分解为可控与不可控两部分的函数

➢ [Abar，Bbar，Cbar，T，k] = ctrbf(A，B，C)

其中　$Abar=\begin{bmatrix} Auc & 0 \\ A21 & Ac \end{bmatrix}$，$Bbar=\begin{bmatrix} 0 \\ Bc \end{bmatrix}$，$Cbar=[Cuc\ \ Cc]$，

(Ac，Bc) 是可控的子空间，而 $Cc(sI-Ac)^{-1}Bc=C(sI-A)^{-1}B$。

T 为变换矩阵，Abar = T * A * T'，　Bbar = T * B，　Cbar = C * T'。

k 是长度为 n 的矢量，其元素为各块的秩。

（4）把系统分解为可观与不可观两部分

➢ [Abar，Bbar，Cbar，T，k]=obsvf(A，B，C)

其中　$Abar=\begin{bmatrix} Ano & A12 \\ 0 & Ao \end{bmatrix}$，$Bbar=\begin{bmatrix} Bno \\ Bo \end{bmatrix}$，$Cbar=[0\ \ Co]$，

(Ao，Bo) 是可观的子空间，而 $Co(sI-Ao)^{-1}Bo=C(sI-A)^{-1}B$

T 为变换矩阵，Abar = T * A * T'，　Bbar = T * B　，　Cbar = C * T'。

k 是长度为 n 的矢量，其元素为各块的秩。

➢ 把系统 sys 中可控又可观的部分取出，构成最小实现系统 rsys 的函数。

rsys = minreal(sys)　　生成阶次最小的状态空间或零极增益模型 rsys。

（5）状态空间的均衡实现函数为 ssba 和 balreal，均衡是为了减少矩阵的奇异性，提高运算的精度。

（6）[sysb，T] = ssbal(sys)　　对系统 sys 作相似变换，使变换后系统 sysb 的系数矩阵 [T*A/T，T*B；C/T 0]的行和列有大致相同的范数。

（7）sysb = balreal(sys)　　它是以 Gramian 矩阵为基础的，除了得出均衡的系统 sysb 外，用下述格式

➢ [sysb，G，T，Ti] = balreal(sys) 还可得出可控又可观的对角 Gramian 矩阵 G，可观测其特别小的项与其他项的差别，作为模型是否可降阶的依据。

（8）rsys = modred(sys，elim)　删除状态空间模型中的由下标矢量 elim 指定的状态，使得模型降阶。为了确定 elim，往往要先调用[sysb，G，T，Ti] = balreal(sys)。

【例 8.16】 系统的可控性与可观性及其结构分解

设系统的状态空间方程为

$$\dot{x}=\begin{bmatrix} 2 & 2 & -1 \\ 0 & -2 & 0 \\ 1 & -4 & 3 \end{bmatrix}x+\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}u$$

$$y=\begin{bmatrix} 1 & -1 & 1 \end{bmatrix}x$$

将其作可控性与可观性结构分解。

解：■ 建模

先作可控性结构分解，得出状态方程系数矩阵的秩 r_A 和可控矩阵的秩 r_c。如果 $r_c=r_A$，则系统完全可控；如果 $r_c<r_A$，则系统有 r_A-r_c 个状态不可控，可控阶梯型分解有效。即系统可分为可控与不可控两个部分。类似地，用 obscf 函数可作可观性结构分解。

■ MATLAB程序q816.m

```
A=[-2, 2, -1; 0, -2, 0; 1, -4, 3]; B=[0; 0; 1]; C=[1, -1, 1]; D=0;   % 系数矩阵赋值
s1=ss(A, B, C, D);                                % 构成 LTI 模型
[Abar, Bbar, Cbar, T, k]=ctrbf(A, B, C)           % 化为可控阶梯型
```

```
rA=rank(A)                                    % 状态方程系数矩阵的秩
rc=sum(k)                                     % 判断可控矩阵的秩
```

■ **程序运行结果**

```
Abar =   -2     0     0
         -2    -2     1
         -4    -1     3
Bbar =    0
          0
          1
Cbar =   -1    -1     1
T =       0     1     0
         -1     0     0
          0     0     1
k =       1     1     0
```

系数矩阵的秩 $r_A=3$

可控矩阵的秩 $r_c=\mathrm{sum(k)}=2$

解的结果为

$$\mathrm{Abar}=\begin{bmatrix}\mathrm{Abar11} & \mathrm{Abar12}\\ \mathrm{Abar21} & \mathrm{Abar22}\end{bmatrix},$$

其中 $\mathrm{Abar11}=[-2\quad 0]$, $\mathrm{Abar12}=0$, $\mathrm{Abar21}=\begin{bmatrix}-2 & -2\\ -4 & -1\end{bmatrix}$, $\mathrm{Abar12}=\begin{bmatrix}1\\ 3\end{bmatrix}$

$$\mathrm{Bbar}=\begin{bmatrix}\mathrm{Bbar1}\\ \mathrm{Bbar2}\end{bmatrix}$$

其中 $\mathrm{Bbar1}=0$, $\mathrm{Bbar2}=\begin{bmatrix}0\\ 1\end{bmatrix}$,

$\mathrm{Cbar}=[\mathrm{Cbar1}\quad \mathrm{Cbar2}]$,

其中 $\mathrm{Cbar1}=-1$, $\mathrm{Cbar2}=[-1\quad 1]$

说明系统有一个状态变量不可控。

【例 8.17】 系统的相似变换

设系统的状态空间方程为

$$\boldsymbol{x}=\begin{bmatrix}1 & 0 & 0\\ 2 & 2 & 3\\ -2 & 0 & 1\end{bmatrix}\boldsymbol{x}+\begin{bmatrix}1\\ 2\\ 2\end{bmatrix}\boldsymbol{u}$$

$$\boldsymbol{y}=\begin{bmatrix}1 & 1 & 2\end{bmatrix}\boldsymbol{x}$$

将它变换为两种规范形式。

解：用MATLAB中的canon函数。

■ **MATLAB程序q817.m**

```
A=[1, 0, 0; 2, 2, 3; -2, 0, 1];  B=[1; 2; 2];  C=[1, 1, 2]; D=0;
s1=ss(A, B, C, D);
```

```
disp('约当标准型'), s2=canon(s1, 'modal')
disp('随伴矩阵型'), s3=canon(s1, 'companion')
```

■ **程序运行结果**

```
Warning: Matrix is close to singular. Results may be inaccurate. RCOND = 1.581139e-018.
```

矩阵接近奇异，条件数= 1.581139e-018.，结果可能不准确。

约当标准型

```
a =             x1          x2          x3
        x1      2.00000     0           0
        x2      0           1.00000     0
        x3      0           0           1.00000
b =             u1
        x1      0
        x2      -6.32456e + 017
        x3      6.32456e + 017
c =             x1          x2          x3
        y1      1.00000     -0.31623    -0.31623
d =             u1
        y1      0
```

随伴矩阵型

```
a =             x1              x2              x3
        x1      0               1.42109e-014    2.00000
        x2      1.00000         -3.55271e-015   -5.00000
        x3      8.88178e-016    1.00000         4.00000
b =             u1
        x1      1.00000
        x2      0
        x3      0
c =             x1          x2          x3
        y1      7.00000     13.00000    23.00000
d =             u1
        y1      0
```

8.5　系统的状态空间法设计函数

状态空间设计有极点配置和二次型性能指标最优设计两种方法。

经典 SISO 控制系统设计方法（例如根轨迹法和频率响应法），是设计一个控制器(校正装置)，使该系统的主导闭环极点具有所期望的阻尼比ζ 和无阻尼自然频率ω_n。也就是该经典设计方法只确定主导闭环极点。与此不同的是，状态反馈极点配置方法可确定所有的闭环极点。当然，配置所有闭环极点是有代价的，需有效地测量所有状态变量，其可选的反馈参数不只是输出变量的增益 K，而是全部状态的增益 $\boldsymbol{K}$ 向量。MATLAB 中的函数 acker 和 place

可用以按极点配置解增益向量 $\boldsymbol{K}$，见表 8.9。

表 8.9　控制工具箱中的状态空间设计函数

分　类	函数名称和典型输入变元	功　能
矩阵方程求解	lyap	解连续李雅普诺夫方程
	dlyap	解离散 Lyapunov 方程
	care	解代数黎卡提方程
	dare	解离散代数 Riccati 方程
极点配置和状态估计器	acker(A，B，p)	SISO 系统极点配置
	place(A，B，p)	MIMO 系统极点配置
	estim(sys，L)	生成系统状态估计器
	Reg(sys，K，L)	生成系统调节器
LQ 最优控制	Lqr(A，B，Q，R)	连续系统的 LQ 调节器设计
	dlqr(A，B，Q，R)	离散系统的 LQ 调节器设计
	lqry(A，B，Q，R)	系统的 LQ 调节器设计
	lqrd(A，B，Q，R，Ts)	连续代价函数的离散 LQ 调节器设计
	kalman(sys，Qn，Rn，Nn)	系统的 Kalman 滤波器设计
	kalmd(sys，Qn，Rn，Ts)	连续系统的离散 Kalman 滤波器设计
	lqgreg(kest，k)	根据 Kalman 和状态反馈增益设计调节器

按极点配置选择增益的 MATLAB 命令就是上面所说的 place，它的输入矩阵为 $\boldsymbol{A}$，$\boldsymbol{B}$ 和 $\boldsymbol{E}$（期望的极点矢量），调用方式为

➢ K=place(A，B，E)

或　K=acker(A，B，E)

在控制工具箱中，用于最佳调节器设计的命令还有 lqrd，lqry 和 dlqr 等。

测量所有状态变量是难以实现的。要从测得的部分变量获取系统的全部变量 X，可在系统中配备状态重构器。在确定性系统中，称之为状态观测器；在随机系统中，称之为状态估计器或状态滤波器，例如 kalman 滤波器。

最优调节器指采用状态反馈，使 X 尽快并尽可能准的接近于 0 而不过多耗能的控制系统。其中线性二次型最优控制器占重要地位。因为一方面指标的物理意义较明确；另一方面它在数学上的处理也较简单。

最优估计（滤波）指从带有随机干扰的观测数据中估计出状态变量。其中的线性最小方差状态估计器（也称 kalman 滤波器）占有最重要的地位。这种估计器的状态估值为观测变量的线性函数，优化准则是估计误差的方差阵为最小。

8.5.1　线性平方调节器问题

该受控对象是线性时不变系统

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

今要使系统在 t_f 时的状态达到 x_f，且在调整过程中保证状态变量 $\boldsymbol{x}$ 和控制作用 $\boldsymbol{u}$ 的加权平方误差积分为最小，即

$$J = \frac{1}{2}\int_{t_0}^{t_f}\left(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{N}\boldsymbol{u}\right)\mathrm{d}t = \min$$

为此要加进一个负反馈调节器

$$u(t) = -Kx(t)$$

若 x 是长度为 n 的列向量，u 是长度为 r 的列向量，则 K 为 $r\times n$ 的增益矩阵。现在的问题是求 K 的最佳值。在 t_f 取无穷大时，称为无限长时间状态调节器问题，此时 K 为常数矩阵。

若 N=0，经过最优化的推导，这个命题最后归结为求解一个代数黎卡提方程

$$PA + A^{T}P + PBR^{-1}B^{T}B - Q = 0$$

其中未知数只有方差阵 P。求得 P 阵以后，就可按下式求得 K

$$K = Q^{-1}B^{T}P$$

在 MATLAB 控制工具箱中有解代数黎卡提方程的专用函数 are(Algebra Ricatti Equation)。对于形式为 $A'X + XA - XBX + C = 0$ 的代数黎卡提方程，它的调用方式为

➢ X = are(A，B，C)

显然在本问题中，X 是 P，A 是 A，代入 B 的应是 $BR^{-1}B$，代入 C 的应是 Q。

可以不先求中间解 P 而直接求 K。MATLAB 给出了解决线性平方估计器的函数 lqr，(Linear Quadratic Regulator)，其调用语句为

➢［K，P，E］= lqr(A，B，Q，R，N)

可见其输入变元都是系统中的已知矩阵，而返回的解除了增益 K 和方差阵 P 之外，还有一个特征根矩阵 E，它是特征方程 $\lambda I-(A-BK) = 0$ 的根，根据它可以判断控制器的动态响应及稳定性。

最佳线性平方调节器的设计公式相当严整，但是用起来也有困难，主要是权重矩阵 Q 和 R 的取法带有很大的主观任意性。取得不好设计出来的系统并不满意。因此，人们往往宁愿退到确定性控制的概念上去，即保证系统的闭环极点具有较高的频率和适当的阻尼系数。也就是按照极点位置来选择增益，而不去解复杂的黎卡提方程。

8.5.2 线性平方估计器问题

调节器是控制问题，估计器则是观测问题。其目标是在输入干扰和测量干扰下最准确地估计出系统的状态。因此从模型上就不能用确定性模型，要考虑有随机输入的系统，在不考虑控制问题时，系统的方程可简化，表为

$$\dot{x} = Ax + Gw$$

$$z = Cx + v$$

其中 w，v 分别为满足下列条件的随机输入干扰和测量干扰。

$$E(w) = E(v) = 0,\ E(ww^{T}) = Q_n,\ E(vv^{T}) = R_n,\ E(wv^{T}) = N_n$$

另外，设计的目标函数也不同。随机系统计算与确定性系统计算的主要区别在于，它要求的是输出误差的统计特性，而不是求某一给定 $u(t)$，$w(t)$，$v(t)$下的输出波形。随机系统的参数通常是按照系统的某种误差（或变量）以时间平方积分（或方差）为最小的准则进行设计，在这里是要根据测量向量 $z(t)$与它的估计值 $\hat{z}(t) = C\hat{x}(t)$ （三角帽表示估计值)之间的误差，加一个负反馈 $\dot{\hat{x}} = A\hat{x} + L(z(t) - C\hat{x}(t))$，问题是要选择适当的 L 值，使 x 的测量估计值 $\hat{x}$ 与实际值 x 的误差均方值

$$P = \lim_{T\to\infty} E\left[(x(t) - \hat{x}(t))^{T}(x(t) - \hat{x}(t))\right] = \min$$

为最小。这等价于一个滤波系统。这个数学系统称为最小方差观测器或估值器。

经过比较冗长的推导，可以证明，这个问题也归结为解下述黎卡提方程

$$AP + PA^{T} - PC^{T}R_n^{-1}CP + GQ_nG^{T} = 0$$

在这个方程中，只有 P 是未知矩阵，因而是可解的。求出 P 后，就可求出最佳的增益 L。

$$L = PC^{T}R_n^{-1}$$

P 和 L 就是时不变稳态卡尔曼滤波问题的解。

可以看出，它与最小平方误差控制器在数学上有对偶性。如果调用解黎卡提方程的函数

X=are(A，B，C)

则 X 应代以 P，A 仍为 A，B 应代以 $C^{T}R_n^{-1}C$，C 应代以 GQ_nG^{T}。同样也可以不先求中间解 P 而直接求 L。解线性平方估计器的函数为 kalman，其调用语句为

➢ [kest，L，P] = kalman(sys，Qn，Rn，Nn)

可见其输入变元 A，B，C 都已包括在 sys 中，而返回的除了增益矩阵 L 和方差矩阵 P 之外，还有滤波器的状态空间 LTI 模型 kest。根据它可以直接求状态变量的估计值，也很容易判断滤波器的动态响应及稳定性。

函数 kalman 也适用于离散对象，因为对象的离散特征已包含在 sys 中。在离散对象的情况，输出变元还可增加。

最优观测器实现的困难往往在于难以获得准确可靠的干扰数据，即 Q_n，R_n，N_n 很难得到。这时人们可按估值器的特征方程根 E 的配置来求出增益 L 的值，即不求“最优”而求“较好”。极点配置法的函数与调节器相同，只是输入变元有所不同。其调用命令为

➢ L = place(A'，C'，E)'

或　L = acker(A'，C'，E)'

其中 E 为估值器的特征根的预定值向量。

用任何一种方法求得 L 和（或 K）后，系统的 LTI 模型可由 estim 或 reg 函数求出。

【例 8.18】 全状态反馈极点配置设计

设系统的状态方程为

$$\dot{x} = Ax + Bu$$

其中 $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -5 & -6 \end{bmatrix}$，$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

要求利用状态反馈控制 $u = -Kx$，将此系统的闭环极点配置成 $p_{1,2} = -2 \pm j4$ 及 $p_3 = -10$。求状态反馈增益矩阵 K。

解：■ 建模

极点配置设计的一般步骤如下。

（1）检查并确认系统的可控性，否则无解。用条件 rank(ctrb(A，B)) = n。

（2）求现有开环系统特征多项式的系数向量α = poly(A)，注意，它的长度为 n+1。

（3）求将系统变换为标准型的变换矩阵。

T = ctrb(A，B)*L

$$其中\quad L=\begin{bmatrix}\alpha_n & \alpha_{n-1} & \cdots & \alpha_2 & 1\\ \alpha_{n-1} & \alpha_{n-2} & \cdots & 1 & \\ & \cdots & \cdots & & \\ \alpha_2 & 1 & & & 0\\ 1 & & & & \end{bmatrix}$$

（4）求系统预期闭环特征多项式的系数向量β = poly(p)。

（5）状态反馈增益矩阵 $K = dab*T^{-1}$

其中　　$dab = [(\alpha_{n+1}-\beta_{n+1}),\ (\alpha_n-\beta_n),\ \cdots,\ (\alpha_2-\beta_2)]$

将整个过程组成程序包，它的输入变量应为 A，B 和 p，而待求的输出则为状态反馈增益矩阵 K。函数 place 和 acker 就集成了这个过程，不过具体的算法不同。

■ MATLAB程序q818.m

```
A=[0, 1, 0; 0, 0, 1; -1, -5, -6]; B=[0; 0; 1];
p(1)=-2+2i; p(2)=-2-2i; p(3)=-10;
% 方法 1，按五个步骤做
Co = ctrb(A, B); n=size(A);
if rank(Co)<n error('系统不可控，无解')
else
   alpha = poly(A); n=rank(A);
   L = hankel([alpha(n:-1:2)'; 1]);
   T = Co*L;
   beta=poly(p);
   K = (beta(n+1:-1:2)-alpha(n+1:-1:2))/T
end, pause
% 方法 2，用 MATLAB 现成函数 place 及 acker
Kp = place (A, B, p)
Ka = acker(A, B, p)
```

■ 程序运行结果

```
K =     79.0000    43.0000     8.0000
Kp =    79.0000    43.0000     8.0000
Ka =    79         43          8
```

【例 8.19】 连续系统状态观测器设计

设系统的状态方程为

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -6 & -11 & -6\end{bmatrix},\quad B=\begin{bmatrix}0\\0\\1\end{bmatrix},\quad C=[1\ 0\ 0]$$

要求设计全阶状态观测器 $\dot{\hat{x}} = A\hat{x} + L(z(t) - C\hat{x}(t))$，使它的闭环极点配置成 $p_{1,2}=-2\pm j2\sqrt{3}$，$p_3=-5$。求状态观测增益矩阵 L。

解：■ 建模

极点配置设计状态观测器是控制器的对偶，其设计步骤相仿，只是把 ctrb 换成 obsb，或者把 $\boldsymbol{A}$ 换成 $\boldsymbol{A}'$，$\boldsymbol{B}$ 换成 $\boldsymbol{C}'$。本题采用直接调用程序包 place 和 acker 的方法，它的输入变量应为 $\boldsymbol{A}$，$\boldsymbol{C}$ 和 $\boldsymbol{p}$，而待求的输出则为状态观测增益矩阵 $\boldsymbol{L}$。此时函数 place 和 acker 的输入变元为 $\boldsymbol{A}'$，$\boldsymbol{C}'$ 和 $\boldsymbol{p}$。即 L= place(A'，C'，p)' 或 L= acker(A'，C'，p)'。

由原系统与状态观测器构成的整个新系统模型的方程为

$$\dot{\boldsymbol{x}}_e = [\boldsymbol{A}-\boldsymbol{LC}]\dot{\boldsymbol{x}}_e + \boldsymbol{L}\boldsymbol{y}$$

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{x}_e$$

可用函数 estim 求得，即 est=estim(sys，L)。注意此模型中的 $\boldsymbol{x}_e$ 为系统原变量 $\boldsymbol{x}$ 的估计值，$\boldsymbol{y}_e$=$\boldsymbol{x}_e$。

■ MATLAB程序q819. m

```
clear, format compact
A=[0, 1, 0;0, 0, 1;-6, -11, -6];B=[0;0;1];C=[1 0 0];D=0;
p(1)=-2+2*sqrt(3)*i;p(2)=-2-2*sqrt(3)*i;p(3)=-5;
Lp = place (A', C', p)'        % 用工具箱函数 place 及 acker
La = acker(A', C', p)'
% 求此状态观测估计器与原系统合成的模型
disp('原系统的模型')
s1=ss(A, B, C, D);        % 原系统的模型
disp('合成系统的模型')
es1=estim(s1, Lp);           % 合成系统的模型
```

■ 程序运行结果

```
Lp = 3.0000
     7.0000
    -1.0000
La = 3.0000
     7.0000
    -1.0000
```

为节约篇幅，此处不列出原系统的模型 s1 和合成系统的模型 es1。读者可自行实践。要注意显示出的系统状态方程中，y1 为原系统的 y，y2=x1，y3=x2，y3=x3 分别为原系统状态变量[x1;x2;x3]的估计值[xe1;xe2;xe3]。

【例 8.20】 离散系统状态观测器设计

设离散系统的状态方程为

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_d\boldsymbol{x}(k) + \boldsymbol{B}_d\boldsymbol{u}(k)$$

$$\boldsymbol{y}(k) = \boldsymbol{C}_d\boldsymbol{x}(k)$$

其中 $\boldsymbol{A}_d = \begin{bmatrix} 0 & 0 & -0.5 \\ 1 & 0 & 0.2 \\ 0 & 1 & 0.8 \end{bmatrix}$，$\boldsymbol{B}_d = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$　$\boldsymbol{C}_d = [1\ 0\ 0]$

要求设计全阶状态观测器，使它的闭环极点配置成 $p_{1,2}=-0.5\pm0.7\mathrm{j}$，$p_3=-0.1$。

解：■ 建模

极点配置设计离散系统状态观测器的步骤和调用函数与连续系统相仿。此时函数 place 和 acker 的输入变元为 Ad'，Cd'和 pd。即

Ld = place(Ad'，Cd'，pd)'

或　Ld = arker(Ad'，Cd'，pd)'

此题中把系统原有极点和配置后的极点都画在 z 平面上作比较。

■ MATLAB程序q820.m

```
clear,
Ad=[0, 0, -0.5;1, 0, 0.2;0, 1, 0.8];      % 设定系统原有参数
Bd=[0; 0; 1];Cd=[1, 0, 0];
pbd=[0.5+0.7j, 0.5-0.7j, -0.1];         % 要求实现的状态观测器极点
Lp=place(Ad', Cd', pbd)                  % 用两种函数来求
La=acker(Ad', Cd', pbd)
pd=eig(Ad);                              % 求系统原有极点
% 画 z 平面上零极点分布, 查 help zplane 知零极点均用列向量做变元, 无零点时输入 NaN。
figure(1), zplane(NaN, pd)               % 原有零极点
figure(2), zplane(NaN, pbd')             % 把极点 pbd 化为列向量画图
```

■ 程序运行结果

如图 8.16 所示。

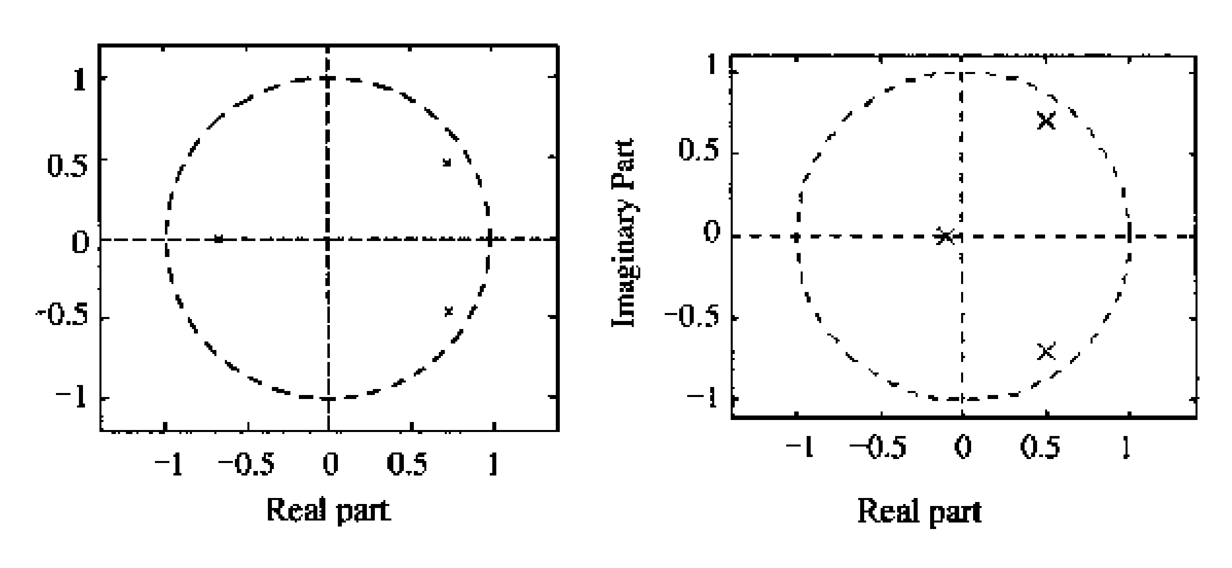


图 8.16　原系统（左）和配置状态观测器后的系统极点分布

【例 8.21】 二次型最优调节器的设计

设系统的状态方程为

$$\dot{x} = Ax + Bu$$

其中 $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -35 & -27 & -9 \end{bmatrix}$，　$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

性能指标为　$J = \int_0^\infty (x'Qx + u'Ru)\,dt$

式中　$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$，　$R = 1$

求黎卡提方程的正定矩阵解 $\boldsymbol{P}$，最佳反馈增益矩阵 $\boldsymbol{K}$ 和矩阵 $\boldsymbol{A-BK}$ 的特征值。

解：■ 建模

这是二次型最优调节器的典型问题，可以用控制系统工具箱中 lqr 函数求解。

➢ [K，P，E] = lqr(A，B，Q，R，N)

因为 J 的积分核中，没有 $\boldsymbol{x'Nu}$ 项，$\boldsymbol{N}=0$。由此，很容易写出它的 MATLAB 程序如下。

■ MATLAB程序q821.m

```
clear.
A=[0, 1, 0;0, 0, 1;-35, -27, -9];B=[0; 0; 1];          % 设定系统系数
Q=eye(3); R=1;                    % 设定 Q, R 矩阵
[K, P, E]=lqr(A, B, Q, R)     % 调用 lqr 函数
```

■ 程序运行结果

最佳反馈增益矩阵为

```
K =
   0.0143     0.1107     0.0676
```

黎卡提方程的正定矩阵解为

```
P =   4.2625     2.4957     0.0143
      2.4957     2.8150     0.1107
      0.0143     0.1107     0.0676
```

矩阵 $\boldsymbol{A-BK}$ 的特征值为

```
E =   -5.0958
      -1.9859  +  1.7110i
      -1.9859  -  1.7110i
```

【例 8.22】 加权系数对二次型最优调节器的影响

设系统的状态方程为

$$\dot{\boldsymbol{x}}=\boldsymbol{Ax}+\boldsymbol{Bu}$$
$$\boldsymbol{y}=\boldsymbol{Cx}+\boldsymbol{Du}$$

其中 $\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -1 & -4 & -9\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\0\\1\end{bmatrix}$，$\boldsymbol{C}=[1\ 0\ 0]$，$\boldsymbol{D}=0$

性能指标为 $J=\int_0^{\infty}(\boldsymbol{x'Qx}+\boldsymbol{u'Ru})\mathrm{d}t$

式中 $\boldsymbol{Q}=\begin{bmatrix}q & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$，$\boldsymbol{R}=$标量

试讨论 q 及 $\boldsymbol{R}$ 取值不同对最佳反馈增益矩阵 $\boldsymbol{K}$ 和阶跃响应的影响。

解：■ 建模

这仍然是二次型最优调节器的典型问题，可以用控制系统工具箱中 lqr 函数求解。

➢ [K，P，E] = lqr(A，B，Q，R)

按本题的要求，应该使用户能随意输入权重参数 q 和 R，因此，程序中应放入两个 input 语句。由于要比较它们的阶跃响应，故用 step 语句。对于由状态空间描述的 LTI 系统，只

要左端变量设置正确，用格式

[y, t, x] = step(sys)

可以求出阶跃输入下的系统输出 y 以及全部系统状态 x 随 t 变化的序列数据。为了说明 R 的取值对控制作用 u 的影响，程序中同时求出了 u = K*x'，并画在同一张图上。

■ MATLAB程序q822.m

```
A=[0, 1, 0; 0, 0, 1; -1, -4, -9]; B=[0; 0; 1]; C=[1, 0, 0]; D=0;     % 设定系数矩阵
q=input('q= '); R=input('R= ');              % 输入加权常数
Q=diag([q, 1, 1]);                           % 生成权重矩阵
[K, P, E]=lqr(A, B, Q, R); K, E              % 求最优控制的反馈增益矩阵
A1=A-B*K; B1=B*K(1); C1=C; D1=D;             % 加反馈后的系统矩阵
S0=ss(A, B, C, D);                           % 原系统的 LTI 模型
[y, t, x]=step(s0);                          % 原系统的阶跃响应（含全部状态变量）
figure(1),
plot(t, y, '*', t, x), grid                  % 绘制全部状态变量的阶跃响应曲线
s1=ss(A1, B1, C1, D1);                       % 加反馈后系统的 LTI 模型
[y1, t1, x1]=step(s1); u1=K*x1';             % 加反馈后系统的阶跃响应，并求出控制量 u1
figure(2), plot(t1, y1, '*', t1, u1, '.-', t1, x1), grid   % 绘制响应曲线
legend('y1', 'u1', 'x1')
```

■ 程序运行结果

输入

```
q=100, R=1,
```

得到

```
K =    9.0499    10.2286    1.1221
E =  -8.6042
     -0.7590 + 0.7694i
     -0.7590 - 0.7694i
```

输入

```
q=10, R=10,
```

得到

```
K =    0.4142    0.8834    0.1031
E =  -8.5514
     -0.2759 + 0.2988i
     -0.2759 - 0.2988i
```

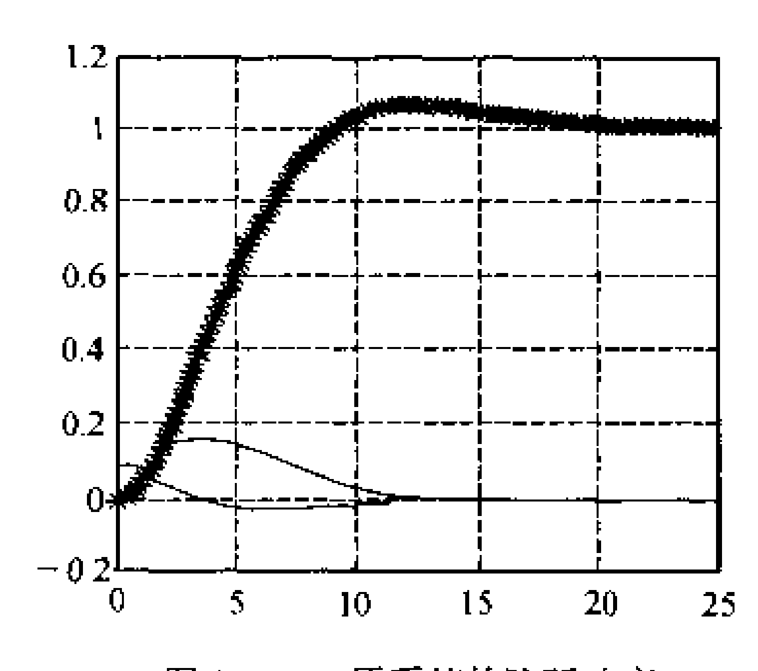


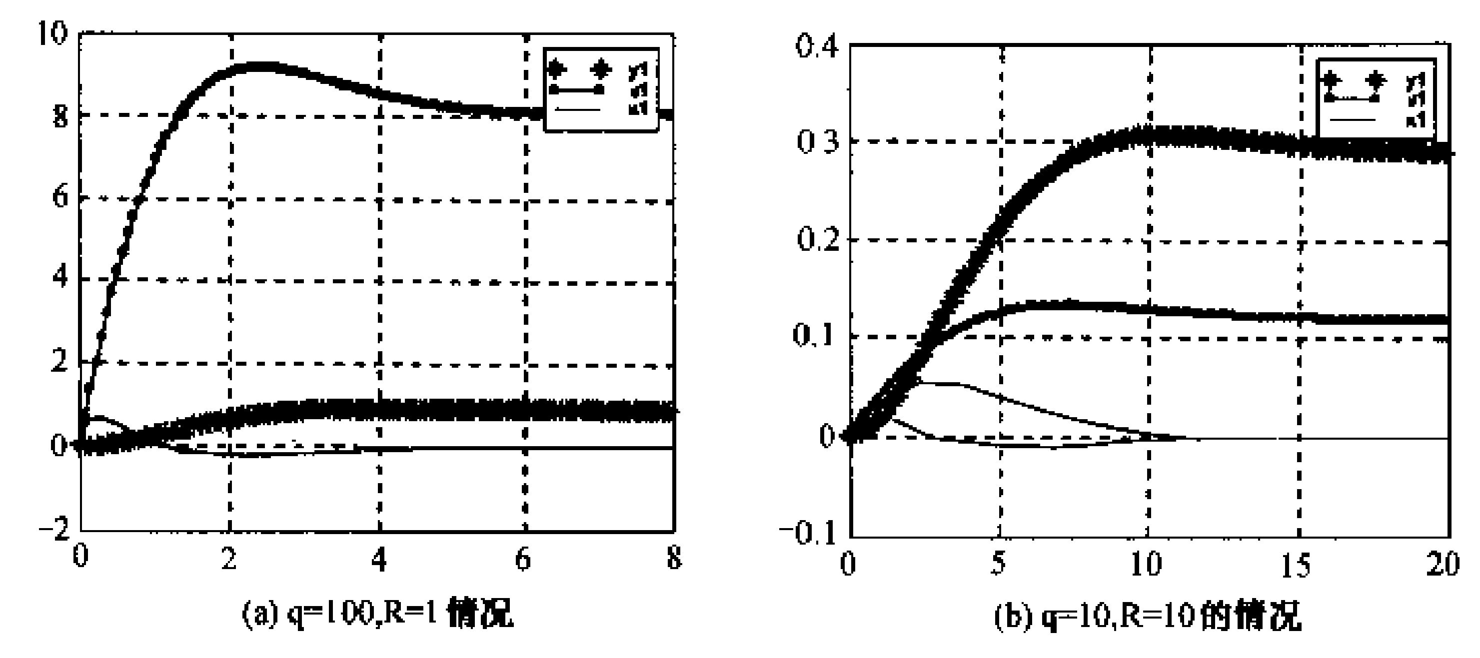
图 8.17-1　原系统的阶跃响应

原系统的阶跃响应画在图 8.17-1，当然它不受权重参数 q 和 R 的影响，两种情况下是一样的。加最优反馈后系统的阶跃响应见图 8.17-2。

在 q = 100，R = 1 的情况下，系统不在乎 u 的大小，它力求以减小 x 中的第一个分量 x(:,1)来减小 J，此时，曲线中的 u1 值就相当大（注意坐标刻度）。同时，过渡过程的时间比原系统大大缩短。由约 20s 缩

短到约 5s。这从特征方程的根上也可看出。

在 q = 10，R = 10 时，u 的大小对 J 的影响大大增加，最优反馈必须避免太大的控制作用。反映在输出曲线中，u1 几乎减小了 50 倍，同时，过渡过程的时间加长了。特别是稳态误差太大了，本来应该为 1，实际输出 y 约为 0.3，在实际系统中，这通常是不能接受的。所以，尽管最优控制的理论很严密，但权重系数的选择还是人为的，与工程实际经验有很大关系。



(a) q=100,R=1 情况　(b) q=10,R=10 的情况

图 8.17-2　加最优反馈后系统的阶跃响应

第 9 章　MATLAB 工具箱简介

本书前 4 章介绍的是 MATLAB 的基本部分，它处在 MATLAB\toolbox\matlab 的子目录中。在比较完整的 MATLAB 专业版的工具箱（toolbox）子目录内实际上还有其他的 20 多个子目录，一般所谓的工具箱，就是指 matlab（注意大小写 MATLAB 和 matlab 的区别）子目录之外的这些子目录，第 7 章介绍的信号处理工具箱和第 8 章介绍的控制系统工具箱各是其中之一。这些工具箱大体可以分为 3 类。

第 1 类是通用性的，它具有独立的开发环境和基础，不是 MATLAB 基本部分的简单扩展。因此，它也具有基础意义，属于这类的有两个，即符号运算（Symbolic）工具箱和框图仿真(Simulink)工具箱。Symbolic 是利用 MATLAB 的界面调用 MAPLE 软件的工具，MAPLE 是加拿大 Waterloo 大学开发的公式推理软件，在国际上开始最早，也最为成功。从 20 世纪 90 年代初 MATLAB4.x 版本开始，Mathworks 公司就与 Maple 合作，推出 Symbolic，到 MATLAB5.x，它更为完善。Simulink 则是用框图来代替算式的 MATLAB 框图界面工具，虽然它全是 Mathworks 公司开发而且是以 MATLAB 的基本部分为基础的，但也有一套独立的程序结构和方法。

第 2 类是专用工具箱，它的绝大部分是以 MATLAB 基本部分为基础编写的子程序集，目的是解决特定的学科领域或应用领域的问题。这些工具箱中的程序语句，没有一条会超出本书的范围。有几个专用工具箱则还必须用到 Simulink。由于商业上的考虑，这类工具箱之间没有横的依赖关系，各个工具箱可以独立选购，只有其基本部分是必须购买的。除信号处理和控制系统工具箱之外，本章将把其余的工具箱作一简单的浏览。

第 3 类则是属于把 MATLAB 程序进行代码变换和实时运行的子程序，这些子程序通常同时需要 MATLAB 和 Simulink 的支持。这是 MATLAB 走向直接工程设计和仿真的重要关键所在，这方面近几年发展很快，有了不少新的函数库。这些函数库不单由 M 文件构成，有些是由 C 或其他可执行代码来实现的，有些还配有第三方公司开发的相关硬件。属于这一类的函数库有 MATLAB Compiler、Real-Time Workshop（RTW）、Stateflow、xPC Target 以及它们的各种代码生成器（Coder）。

9.1　符号数学（Symbolic Math）工具箱简介

顾名思义，符号数学是以符号（如 a，b，c，x，y，z）为对象的数学，区别于以数字为对象的 MATLAB 基本部分。在大学教育中，符号数学是每门课都用到的，因此，专门以不到 100 美元提供给大学生的版本（Student Edition of MATLAB）中就包括了这个工具箱。国外用 MATLAB 介绍科学计算的教科书中，大概有 15%~20%的例题和习题会用到这个工具箱，本书在把这个工具箱用到各课程中时，考虑了以下两方面。

（1）在大学教学中，推理还是一个基本功。而且在大多数大学课程中，也没有太复杂的推理。一般说，把数值计算交给计算机做，绝大多数老师还是能接受的，如果把推理也交给计算机，对教学是否有利，可能会有较大争议。作者目前的感觉是，符号数学工具箱应该

教给学生，但大量使用还是放在科研中比较恰当。

（2）Symbolic 工具箱对计算机硬件（包括外存、内存、时钟频率等）的要求比较高，在大部分本科学生的机房内，难于达到。

因为计算机发展很快，第二点会迅速改变，主要还是考虑第一点，慎重一些是分两步，先普遍把大学师生从繁重的数值计算中解放出来，然后再考虑在推理方面作些试验探讨。

9.1.1 Symbolic 工具箱的主要功能

Symbolic 工具箱的主要功能有以下 8 项：

（1）用符号定义各种数学运算和函数（syms，symop）等；

（2）对这些函数式进行代数和三角运算，包括因式分解（factor）、展开（expand）、变量置换（subs）、复合函数（compose）等；

（3）微分和积分运算（diff，int）等；

（4）函数的整理和化简（combine，simplify，simple）等；

（5）可变精度的运算，如可以设置任意多个有效计算位数进行计算（vpa、digits）等；

（6）解方程，包括单变量的代数方程、多变量非线性的联立代数方程（solve）、单变量微分方程、多变量联立微分方程（dsolve）等；

（7）线性代数和矩阵运算（determ，linsolve）等；

（8）变换，包括拉普拉斯变换（laplace）、傅立叶变换（fourier）和 z 变换（ztrans）等。

说明 括号内的英文字符串是 Symbolic 工具箱中的函数名。

9.1.2 符号数学式的基本表示方法

符号数学是对字符串进行运算的，在 MATLAB 中，如果键入

f=3*x^2+5*x+2，或 y=sin(x)

系统会指出，变量 x 无定义，因为它要求 x 必须是一个数；MATLAB 也可以接受形为 f='3*x^2+5*x+2'，或 y='sin(x)'的语句，这时 f 和 y 都是一个字符串，但它没有任何含义。因为它对字符串中的内容不作任何分析。

Symbolic 工具箱必须要能分析字符串的含义，为此，首先要对符号变量作出定义，用语句“x = sym('x');”就定义了 x 是一个字符（串）变量，此后键入的算式 f=3*x^2+5*x+2，或 y=sin(x)就具有了符号函数的意义，连表明字符串的引号都可省略，f 和 y 也自然成为字符（串）变量。

如果一个数学符号表示式中有多个符号，如

z = a*t^2+b*t+c

可以用简化的多个符号变量定义语句放在此表示式的前面。

syms a b c t

注意 只需对算式的右边符号变量（即自变量）作定义，系统会自动把因变量定义为符号变量。

在作符号运算时，比如求这个代数方程的根，就得知道哪个（些）是变量，其余的则是系数，这时可在语句中指定，比如 r = solve('z = 0', t)表示把 t 作为求解的变量。如果不加说明，键入 r = solve('z = 0')，则系统将自动把离 x 最近的那个（些）符号当做变量来求解，

在本例中，自然会选到 t。

表 9.1 以表格方式列出了一些算例，读者可对符号数学工具箱的功能有一个大体的印象。

表 9.1　符号推理的语句及其结果的举例

运　算　式	符　号　式	系 统 响 应
符号函数赋值	r = x^2 + y^2	r =x^2+y^2
符号函数赋值	theta = atan(y/x)	theta =atan(y/x)
符号函数赋值	e = exp(i*pi*t)	e =exp(i*pi*t)
三角函数式化简	f = cos(x)^2 + sin(x)^2	f =cos(x)^2+sin(x)^2
	f = simple(f)	f =1
微分	diff(x^3)	ans =3*x^2
积分	int(x^3)	ans =1/4*x^4
	int(exp(-t^2))	ans =1/2*pi^(1/2)*erf(t)
矩阵按元素群积分	[int(x^a)，int(a^x)，int(x^a，a)，int(a^x，a)]	ans = [x^(a+1)/(a+1)，1/log(a)*a^x，1/log(x)*x^a，a^(x+1)/(x+1)]
解二次代数方程	x=solve('a*x^2 + b*x + c = 0'); x	x = [1/2/a*(-b+(b^2-4*a*c)^(1/2))] [1/2/a*(-b-(b^2-4*a*c)^(1/2))]
解联立代数方程	[u，v] =solve('a*u^2 + v^2 = 0'，'u-v =1')	u = [1/2/(a+1)*(-2*a+2*(-a)^(1/2))+1] [1/2/(a+1)*(-2*a-2*(-a)^(1/2))+1] v = [1/2/(a+1)*(-2*a+2*(-a)^(1/2))] [1/2/(a+1)*(-2*a-2*(-a)^(1/2))]
以 28 位有效数解联立超越方程	digits(28) [x，y]=solve('sin(x+y)-exp(x)*y = 0'，'x^2-y = 2')	x=-6.017327250059306564109729712 y=34.20822723430629650864621443
求一阶微分方程通解	y = dsolve('Dy = -a*y')	y =exp(-a*t)*C1(含任意常数)
给出初始条件求微分方程特解	%y=dsolve('Dy = -a*y'，'y(0) = 1')	y =exp(-a*t)
求二阶微分方程特解，D2 表示二阶导数	% 给出两个初始(边界)条件 y= dsolve('D2y = -a^2*y'，'y(0) =1，Dy(pi/a) = 0')	y = cos(a*t)
求二阶微分方程特解	y =dsolve('(Dy)^2 + y^2=1'，'y(0) = 0')	y =[sin(t)](有两个解) [-sin(t)]
拉普拉斯变换	f=exp(-a*t)*cos(w*t)	f=exp(-a*t)*cos(w*t)
	F=laplace(f)	F=exp(-a*t)*s/(s^2+t^2)
	pretty(F) 此命令用来改善公式可读性	$\frac{\exp(-at)s}{s^2+t^2}$
拉普拉斯变换	f=exp(-a*t)*cos(w*t)	f=exp(-a*t)*cos(w*t)
	F=laplace(f)	F=(s+a)/(s+a)^2+q^2)
	pretty(F) 此命令用来改善公式可读性	$\frac{s+a}{(s+a)^2+w^2}$

注：在表中，假定符号自变量的定义已经给出，则键入第 2 列的符号式就可得到第 3 列的结果

表中最后两个拉普拉斯变换式结果不同。其原因在于前者用 w 表示频率，而后者用 q。因为 w 离 x 比 t 离 x 近，MATLAB 在做拉普拉斯变换时误把 w 当做了自变量，所以结果不对。而后者则把 t 当做自变量，因而结果是对的。为了节省篇幅，这里尽量选了一些简单的推导式，实际上可以推导很繁的式子。目前 MATLAB 符号数学推理的限度是，表示式的长度不超过 1400 个字符。在一般公式推导意义下使用 MATLAB 是很方便的，只是不给自变量赋以数值，而代之以

syms　自变量 1　自变量 2　自变量 3　…

以后的编程和普通 MATLAB 程序完全相同。其执行的结果自然是表达式而不是数值解。如果要做进一步的工作，例如化简、代换、代入数值、解联立方程等等，那就需要对这个工具箱有较完整的了解。

不管在课程中是否使用这个工具箱，应该看到的是计算机科学已经成功地进入了推理领域，已经可以被普通的科技人员在微机上实现。我国数学家吴文俊教授在计算机推理领域做出了国际领先的成果，并因此荣获国家最高科技奖，这也代表了国际科技发展的方向。没有任何理由和方法能够阻止人们去接触、学习、掌握和应用这些推理软件，教师要考虑这一点并探索它对大学教育改革可能产生的影响。教师首先要懂得，要会用。然后应该让学生了解这些软件的功能，便于他们根据自己的需要来选择。因此，本书中在例 6.20 和例 7.14 中做了初步的演示，供读者参考。

9.2　系统仿真（Simulink）工具箱简介

9.2.1　概述

Simulink 是 MATLAB 各种工具箱中比较特别的，一般工具箱只是把面向某一类问题的程序包集中起来，其中的程序都是用 MATLAB 语言编写的，这些工具箱是 MATLAB 在量方面的扩充，而 Simulink 工具箱却是从底层开发的一个完整的仿真环境和图形界面。在这个环境中，用户可以利用鼠标或键盘，完成面向框图系统仿真的全部过程，并且可以更加直观、快速和准确地达到仿真的目标。原来的 MATLAB 是在文本窗口中编程，图形窗口只是用来显示，而 Simulink 则把图形窗口扩展为可以用框图方式来编程，使 MATLAB 的功能有了一个质的飞跃。因此，Mathworks 公司往往把 Simulink 与 MATLAB 并列进行宣传。

Simulink 与一般工具箱不同的另一个优点是它不给出任何新的函数。对用户来说，Simulink 中所用的仍然是 MATLAB 的语法，MATLAB 原有的函数在 Simulink 中仍然有效。因此，读者如果有了 MATLAB 的基础，除了要学一下图形界面的使用方法之外，不需要再学新的语法，马上就会应用。这也显示了其对用户的友好性。

Simulink 作为面向框图的仿真软件，具有以下的功能：

（1）用方框图的绘制代替程序的编写。构成任何一个系统框图有三个步骤，即选定典型环节、相互联接和给定环节参数。这三步可以在一个图形界面上用鼠标和键盘来完成。

（2）仿真的建立和运行是智能化的。首先，画好了框图并存起来，它就自动建立起了仿真的方程；其次，在运行时用户可以不给步长，只给出要求的仿真精度，软件会自动选择能保证给定精度的最大步长，使得在给定的精度要求下系统仿真具有最快的速度。

（3）输入输出信号来源形式的多样化。其输入信号可以是各种信号发生器；也可以来自一个设定的记录文件；还可以来自 MATLAB 的工作空间（workspace）。输出信号也类似，这就扩大了仿真系统与各种外部软件和硬件的接口能力。

9.2.2　环节库及框图的建立

1．环节库及其输入

在 Simulink 中，设有几十种典型环节，分列在信号源（Sources）、输出信号漏（Sinks）、线性（Linear）、非线性（Nonlinear）、离散（Discret）、联接（Connections）等几大类图标子菜单中。要开始建立框图，可在 MATLAB 的命令窗中，键入 simulink，屏幕上会出现两个视窗。一个是上述的几类库图标，另一个是供绘制框图用的空白视窗，如图 9.1。

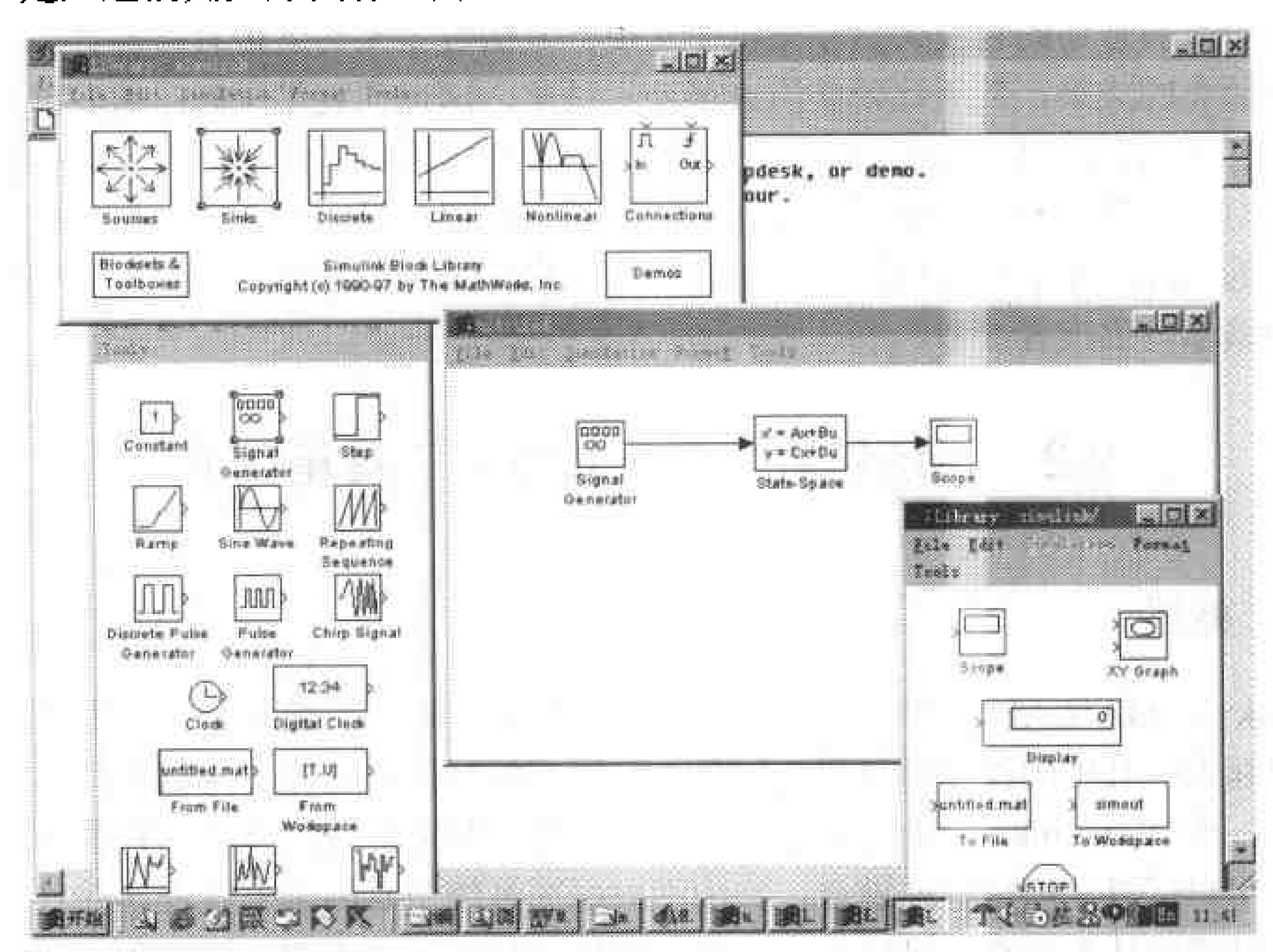


图 9.1　键入 simulink 后出现的仿真框图建立界面

双击 Sources 图标，屏幕上将弹出信号源中包含的所有子图标，如图 9.1 左下方所示。包括常数信号、信号发生器、阶跃信号、斜坡信号、重复序列、时钟信号、随机序列、从文件输入、从工作空间输入等十多种。

双击 Sinks 图标，屏幕上弹出的子图标如图 9.1 右下方所示。其中包括示波器、X-Y 记录仪（X 坐标是状态变量）、数字显示、输出到文件、输出到工作空间等等。

Linear 图标下的库，包含放大环节、纯积分环节、纯微分环节、传递函数模型、零极增益模型、状态空间模型、相加器等等。读者可自行打开其他库来查看其内容，此处从略。

为了建立所需的框图，可以用鼠标器拖曳的方法，将所要的环节移到那个空白的文件中，排列在需要的位置上。

2．环节的联接

把环节都布好后，把各环节的端口按框图联接起来，联接的方法是把鼠标指在线段的

始端，按下左键不放，移动鼠标，一直引到线段的终点端口再释放。此时在终点上将出现箭头。一般环节都只有一个输入端，有些环节如乘法器、逻辑运算等具有双输入端，相加器则可能有更多的输入端，需要先作环节的参数设定，定义输入端的数目。

3. 环节参数的设定

用“双击左键”的方法，逐个打开各个环节的参数设定窗口以修改其中的参数，这些参数可以用 MATLAB 中任何合法的方式表示。

在构成仿真框图时要注意在系统输入端加上信号源，在用户关心的输出端加上信号终端（即观测或记录信号的设备，比如示波器、电压表或者一个文件）。以利于仿真的实施和结果的观测、记录和处理。如果要把时间作为一个输出变量，则必须在框图中加入时钟。

4. 系统的建立

在仿真框图菜单项【File】的下拉菜单中选择【Save】项，把该系统框图赋予文件名后存盘，这时才真正建立了仿真系统。

9.2.3　仿真方法和参数的设定

在仿真框图菜单项【Simulation】的下拉菜单中选择【Parameter】项，此时将出现如图 9.2 所示的仿真参数菜单。其中右边的下拉菜单可选项包括数值积分的 6 种方法（ode45、ode23、ode113、ode15s、ode23s 和全离散），左边的下拉菜单可选项有定步长或变步长，选变步长时必须规定数值积分的相对精度（缺省值为 0.001）和绝对精度（缺省值为 10^{-6}）。必要时还可限定最大和初始积分步长。特别要注意设定仿真的起始和终止时间。如果不对仿真方法和参数进行设定，系统将按其缺省值进行仿真。

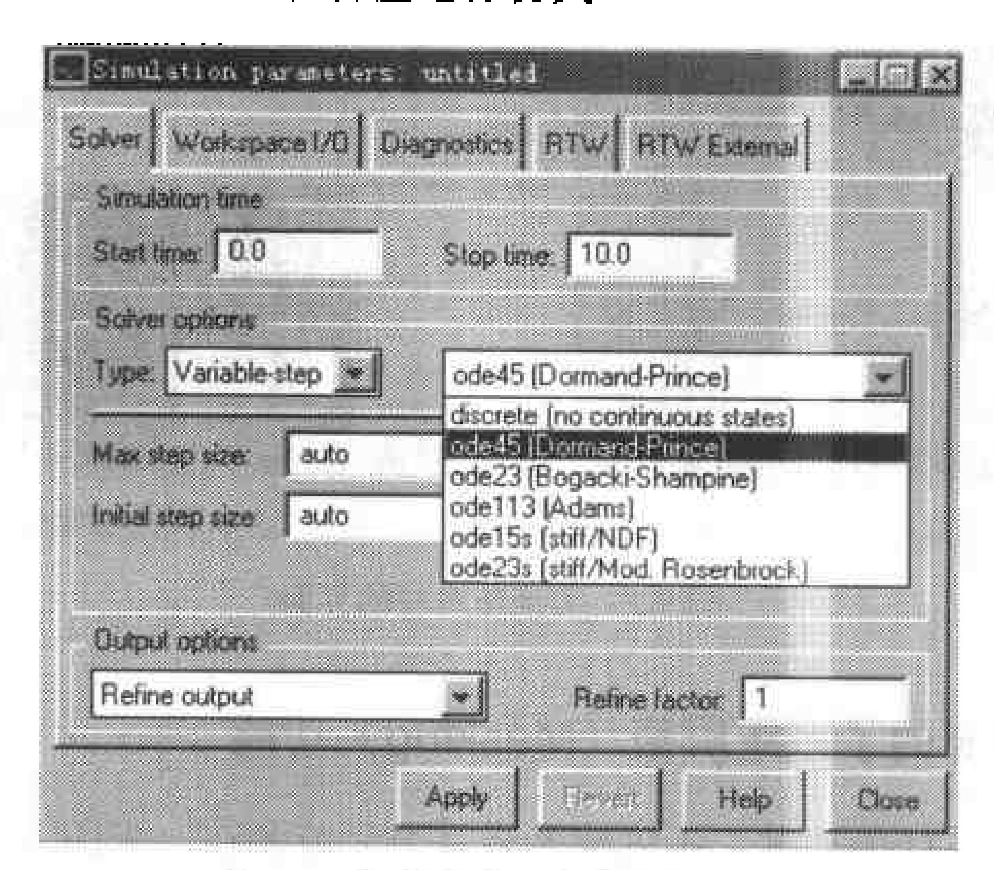


图 9.2　仿真方法和参数设定界面

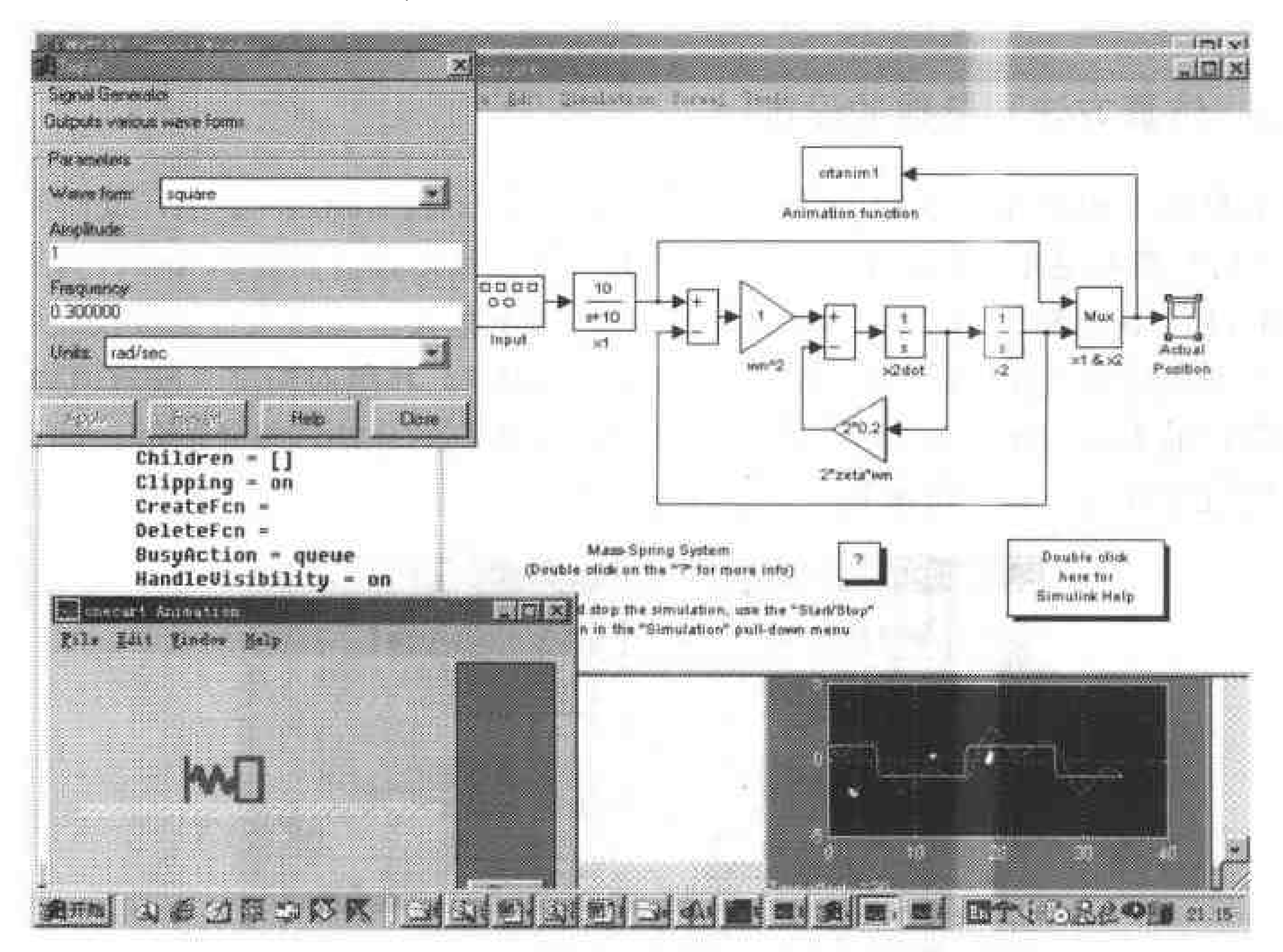
9.2.4　仿真的运行

在仿真框图菜单项【Simulation】的下拉菜单中选择【Start】项，系统就开始仿真。在输出的仪表上可以看到输出曲线。同时 Start 项就变为 Stop，在任何时候单击它，仿真就会

停止。为了全面观察和处理系统运行的结果，可以在多个观察点上设置示波器模块，也可以在一个观察点上同时接上示波器和工作空间模块；这时既可看到曲线的大体形状，又可用 MATLAB 程序对送来的数据直接进行处理。也可以将结果送给一个文件保存起来，以便事后处理。要把两个或多个过程显示在一个坐标系内，可以让它们通过一个多路器（connections 中的 Mux 环节）接到示波器上去。

如果发现错误，或对仿真的结果不满意，可修改框图或修改参数，重新仿真。各种窗口的大小都是可以调整的，窗口的切换用【Alt】+【Tab】组合键，这些都和 Windows 兼容，只要熟悉 Windows，就不会有任何困难。

把 Simulink 和 MATLAB 的动画仿真的能力相结合，可以在仿真的同时显示动画。运行 Simulink 的 demo 程序，可以看到普通两截摆、倒立摆等实体模型运动动画，也可看到质量-弹簧构成的结构振荡运动演示，这对于物理概念的理解很有帮助。图 9.3 是在 demo 演示库中简单的质量-弹簧系统的仿真框图，在仿真进行时可同时看到左下角的动画和右下角的输入输出曲线。这个仿真框图上的输出没有接文件或工作空间，因此，得不到记录的数据。



左上方为信号源参数设定视窗，左下方为动画演示视窗，右下方为示波器输出视窗

图 9.3　实施仿真时的界面

9.2.5　Simulink 的子系统屏蔽（Masking）功能

对复杂的大系统进行仿真时，不可能把成百个环节画在一张图上。这时要求能把系统分级包装为模块或子系统，这些模块或子系统也是由框图组成的。Simulink 提供了创建用户自定义模块的功能。它有如下的一些特点：

（1）把子系统中的多个对话框集中成为一个。因此，改变系统参数时就不必逐个去打开子系统中的各个环节，这种功能称为屏蔽。经过屏蔽的子系统就成为一个黑盒子，可以从外部改变其参数而不涉及其内部结构。

（2）允许用户为该子系统创建自己喜欢的参数修改对话框。

（3）允许用户为该子系统创建自己定义的图标，建立相应的帮助文本。

（4）避免由于偶然因素而破坏或改动子系统的参数，同时也有知识产权保密作用。

这些功能大大方便了 Simulink 的可扩展性，用户只要根据自己的专业领域，编出特定的模块工具箱，就可以利用 Simulink 进行系统的仿真。9.4 节中将介绍的各种模块工具箱都是这样开发出来的。

9.2.6 Simulink 内部工作过程简介

用 Simulink 进行仿真主要经过两个阶段，第一阶段是用图形方式来画结构图，并进行编辑；第二阶段是进行运算仿真。前一阶段所采用的数据结构要便于以图形表示动力学系统，它并不适合于仿真运算，因此，进行仿真时第一件事就是要产生最适于仿真的数据结构，这里包括几个操作步骤。首先，MATLAB 会对每一个模块的参数进行评估，并把作为模块参数的变量或函数换成数字，如果在这步之后用户再去改变框图中的参数，它将不会对仿真结果产生任何影响。其次，模块将按照它们的状态数据更新的先后进行分类排列，在这阶段中首先要把框图展平，即取消各个中间的子系统而代之以全部的框图结构。为了提高速度，所有非数字的框（如文字注释、端口、子系统）全部被取消。如果一个框图在某一计算步长时的输出要依赖于它的输入（例如一个放大器），那它必须要在输入数据更新之后才能更新其输出，这样就可找到仿真运算的排序。如果在求某一环节的输出时，必须知道它本身的值，那就表明出现了代数环路（algebraic loop），这时在每一步长的计算中都要进行迭代运算来求解这个代数方程组，使仿真速度大大降低。

最后要检验各模块之间的联接，以保证每一模块的输出向量的长度（维数）与被它驱动的模块所要求的输入向量一致。建立了仿真的数据结构以后，就可以准备仿真运算了，仿真是用数值积分法进行的。每一个积分器的输出将取决于 Simulink 所建的模型的性能。这实际上就是靠模型的 S 函数（下面再解释）来提供的系统中所有状态变量的导数。这些导数的计算也分两步，首先按照初始化阶段排出的次序计算每一个模块的输出，然后，按照当前时间的数据计算每一个积分模块的导数值。由此得出的导数向量被数值积分器用来算出新的状态向量，一旦得到新的状态向量，采样数据模块和示波器模块的数据也就更新了。

现在再来介绍一下 S 函数，当用户建立了一个 Simulink 模型后，在 MATLAB 中也同时建立了一个函数，这个函数定义了系统的动力学，并且在求积分、线性化等过程中都要用到它。它和其他 MATLAB 中的函数相仿，其调用语句为

➢ sys=model(t,x,u,flag)

其中 model 是模型名称而 flag 用来控制返回的 sys 中的信息。例如 flag=1 将返回系统全部状态变量的导数（在 t 时刻及给定的 x，u 下）。S 函数的这种数学特性也可以用 MATLAB 语言编成的 M 文件来模仿，或用 C 或 Fortran 子程序经编译后形成的 MEX 文件来模仿，S 函数最主要的特点在于，它是通过方框图的图形界面自动生成的。又能以很高的效率进行仿真运算，不少实例说明，它的运算速度比 M 文件要高出将近一个数量级，和目标码 MEX 文件的执行速度相近。

可见 S 函数是一个从框图图形界面转为仿真数据结构的中间桥梁，Simulink 靠它获得了既有优良人机交互，又有较高的仿真速度的特点。

9.2.7　Simulink 应用范围的扩展

Simulink 开始是为控制系统仿真而开发的，以后发现它不仅编程特别方便，不易出错，而且可以解决 MATLAB 程序不好解决的非线性、变系数系统等问题。因此，许多领域都针对本身的特点，开发了各种仿真环节或模块作为工具箱，加入到 Simulink 中去。

在打开 Simulink 时出现的大类图标菜单中，左下角有一个 Block and Toolboxes 方框。它列出了各领域开发的仿真环节库。目前有通信（Comm）、控制系统（Control）、数字信号处理（DSP Blockset）、定点处理（Fix-Point Blockset）、非线性控制（NCD Blockset）、电力系统（Powersys Blockset）、状态流（StateFlow）、系统辨识（System ID Blocks）、其他（Simulink Extras）等等。有了这些库，就很容易用 Simulink 来解决它们的问题，而且只画框图，无需编程。

本书中的电路分析可以使用 Powersys Blockset，因为用到二极管、可控硅、非线性铁磁材料等各种非线性器件的电力电子系统，只有靠 Simulink 才能解决问题。数字信号处理可以用 DSP Blockset 和 Fix-Point Blockset，控制系统（特别是非线性系统）用 Simulink 更是无疑的。通信系统仿真用 Comm 工具箱也特别方便，不过这个问题更专业化，在本书中不可能涉及了。

9.3　以 matlab 为基础的工具箱简介

在 MATLAB\toolbox\子目录下，除去已经介绍过的 matlab、Symbolic 和 Simulink 3 个通用工具箱外，还有以下一些专用工具箱，其中大部分只要以 matlab 工具箱为基础就能运行，少数则还要求有 Simulink 通用工具箱的支持。在大学本科教学中，可能用到的是信号处理和控制系统，本书第 7 章、第 8 章已对这两个工具箱作了详细的介绍。这一节将列出以 matlab 工具箱为基础的其他工具箱的名称和内容，那些还要用到 Simulink 的模块工具箱将在 9.4 节中对其模块组成作进一步说明。读者在需要时，可以查阅 MATLAB 的说明书。国内近年来也出版了不少这方面的书籍可供参阅。

表 9.2　MATLAB 中的专用工具箱

序　号	工具箱名	内　容	注　释
1	comm	通讯系统	还含有 Simulink 用的模块库
2	control	控制系统分析	
	database	数据库	
3	dspblks	数字信号处理框图模型库	全为 Simulink 用的模块库
	Excel Link	与 Excel 联接	
4	fdident	频域辨识	
5	finance	财务	
6	fixpoint	定点运算	全为 Simulink 用的模块库
7	fuzzy	模糊集合	还含有 Simulink 用的模块库
8	hosa	高阶频谱	
9	ident	系统辨识	还含有 Simulink 用的模块库
10	images	图像处理	

续表

序号	工具箱名	内容	注释
11	lmi	线性矩阵不等式设计	
12	map	地理信息	
13	mpc	模型预测控制	还含有 Simulink 用的模块库
14	mutools	μ 分析综合工具	
15	nag	数值分析和统计	
16	ncd	非线性控制	全为 Simulink 用的模块库
17	nnet	神经元网络	还含有 Simulink 用的模块库
18	optim	最优化	
19	pde	偏微分方程	
20	powersys	电力系统	全为 Simulink 用的模块库
21	qft	定量反馈理论	
22	robust	鲁棒性控制	
23	rtw	实时系统设计	需要 Simulink 做基础
24	signal	信号处理	
25	splines	样条函数	
27	stats	数理统计	
28	tour	导游演示	
29	wavelet	小波变换	

9.4 以 Simulink 为基础的模块工具箱简介

这类工具箱主要开发了在该领域的各子系统的模块，使用户无需编程，直接在 Simulink 环境下调用仿真，从而分析设计该类系统。本节只简单介绍它们包括些什么模块，以便读者查找。

在 MATLAB 命令窗内键入 simulink，将出现列举其中环节大类库名的视窗。其左下角有一个模块和工具箱（Blocksets and Toolboxes）方框，双击打开它，可以看到所有含有仿真模块库的工具箱名。在 MATLAB 5.3（R11）版本中，它列出 9 个工具箱。Simulink，Communications Blockset，Control System Toolbox，DSP Blockset，Nonlinear Control Design Blockset，Power System Blockset，Stateflow，Simulink Extras，System ID Blocks。2000 年新出的还有 CDMA Reference Blockset，Dial & Gauge Blockset 等。其中 Simulink 和 Control System Toolbox 已在 9.2 节中介绍过，NCD Blockset 和 Simulink Extras 只是在控制方面增加了一些非线性环节和其他环节，没有新的概念。Stateflow 和 System ID Blocks 超出了本书的范围。因此，本节对其他 4 个模块库的内容只作简略的介绍。

9.4.1 电力系统（Powersys）模块工具箱简介

电力系统工具箱（Powersys）提供了电力传输和拖动中用到的各种子系统模型，它包含了以下的 7 大类库，含有数十种模块模型。

（1）电源——包括交流电流源、交流电压源、直流源、可控电流源、可控电压源等。

（2）元件——包括断路器、变压器、互感、长线、饱和变压器、串联 RLC 电路、并联 RLC 电路、浪涌吸收器等。

（3）电机——同步电机、异步电机、永磁同步电机、水力涡轮调速机等，另外包括可单独提取这些电机运转参数的各测量分路器。

（4）电力电子——闸流管、二极管、GTO、Mosfet、理想开关等。

（5）测量——电流测量、电压测量等。

（6）联接——零线、L 联接器、T 联接器、总线、地线等。

（7）其他——控制模块、三相子库、直流电机、定时器等。

9.4.2 数字信号处理(DSP Blocks)模块工具箱简介

数字信号处理模块工具箱提供了与数字信号处理工具箱中的各种函数相对应的 Simulink 模块。分为 7 大类库，含有上百种模块模型。

（1）信号源库——复数常数、复数常数矩阵、来自工作空间、来自文件窗函数、*n* 点采样使能等。

（2）信号漏（终端）库——时间向量显示器、频率向量显示器、复数 FFT 显示器、去工作空间的复数矩阵、去工作空间的实数矩阵等。

（3）通用 DSP 库，它分为以下 5 个子库。

① 信号运算子库——延迟、补零、加窗、减少采样率、增加采样率等。

② FFT 变换。

③ 缓冲子库——缓冲器、去缓冲器、部分缓冲器等。

④ 开关和计数器子库——*n* 点采样、交换器、分配器等。

⑤ 与 Simulink 联接子库——多路器、分路器、读写文件、触发信号等。

（4）数学函数库，分为以下几种子库。

① 初等数学子库——以下还有十几个模块。

② 矩阵运算子库——以下还有十几个模块。

③ 复数运算子库——以下还有十几个模块。

④ 统计运算子库——以下还有十几个模块。

（5）滤波器库，分为以下几种子库。

① 滤波器设计子库——模拟滤波器设计、数字 IIR 和 FIR 滤波器设计、各种其他滤波器设计等。

② 滤波器实现子库——滤波器、多通道滤波器、交迭相加滤波器等。

③ 自适应滤波——LMS，RMS，Kalman 滤波等。

④ 多采样率滤波子库——抽取、内插、变换采样率等。

（6）频谱分析库——周期图。

（7）演示库。

9.4.3 定点处理（Fix-Point Blocks）模块工具箱简介

定点处理是为了分析计算机字长有限对处理信号造成的影响。因为这是一个非线性问题，很难用解析方法得出准确的结果。因此，采用 Simulink 仿真成为主要的手段。定点处

理模块工具箱提供了定点处理的各种 Simulink 模块，它有几十种模块模型，没有再分子库。它的分类如下。

- 定点的算术运算（加、减、乘、除、累加等）；
- 定点的逻辑运算（与、或、非、异或、关系运算等）；
- 定点的变换运算（一维和二维查表、定浮点变换、定界、饱和等）；
- 定点数的传送（来自定点、去往定点、去往工作空间、来自工作空间等）；
- 定点数的联接（多路器、分路器、开关等）。

在考虑处理器字长有限的情况下将实际的数字信号处理系统画成框图，调用定点处理模块，构成 Simulink 仿真结构图，就可以通过仿真比较它与无限字长条件下的结果有何差别。

9.4.4　通信系统（Comm）模块工具箱简介

在 MATLAB 命令窗环境下，键入 commlib，屏幕上会出现以下的通信系统结构图，通信系统模块工具箱就是按照这个结构图组织它的库函数的。

在任何一个方框上用鼠标双击，就可以显示该环节有多少种类型可供选择，选择好类型后又可进一步选择参数。把每一个环节的类型和参数确定以后，该通信系统也就确定了。下面的问题是进行仿真、确定系统的性能指标。并在各种方案之间进行选择比较。

这里只对其中的库函数作一些简单的列表，因为发和收是成对偶关系，所以每一对偶共用一个函数库，所以在主通道的 13 个方框中，共 7 个库。

（1）信号源和信号终端库——Simulink 中的信号源和信号漏（终端）、另加脉冲触发读/写工作空间、读/写文件、采样同步脉冲生成、眼图显示、误码率检测、示波器、各种分布律的噪声发生器等。

（2）信源编码和信源解码库——采样量化编码/解码、差分脉冲编码调制（DPCM）编码/解码、μ-律压缩/扩张器、A-律压缩/扩张器等。

（3）纠错编码和纠错解码库——Hamming、BCH、R-S、循环、线性、卷积编码/解码器。

（4）调制和解调库，有以下两类子库。

① 模拟调制和解调子库——双边带调幅（DSB-AM）、正交调幅（QAM）、调频（FM）、调相（PM）、单边带调幅（SSB-AM）等。

② 数字调制和解调子库——多电平振幅键控（MASK）、同步-正交边带振幅键控（S-QSAK）、相干-正交边带振幅键控（C-QSAK）、异步-正交边带振幅键控（A-QSAK）、最大频移键控（MFSK）等。

（5）多路存取库——数字时分多路（TDMA）/分路器、时分多路/分路器、频分多路/分路器、码分多路/分路器等。

（6）发送滤波和接收滤波库——R-C 滤波、Sinc 滤波、Hilbert 滤波等。

（7）信道库，有以下几类子库。

① 通带信道——各种干扰和噪声；

② 基带信道——各种干扰、噪声和衰落。

以上是主信道上的库，其他还有一些附属的工具库和演示库，读者可自行打开查阅，此处从略。

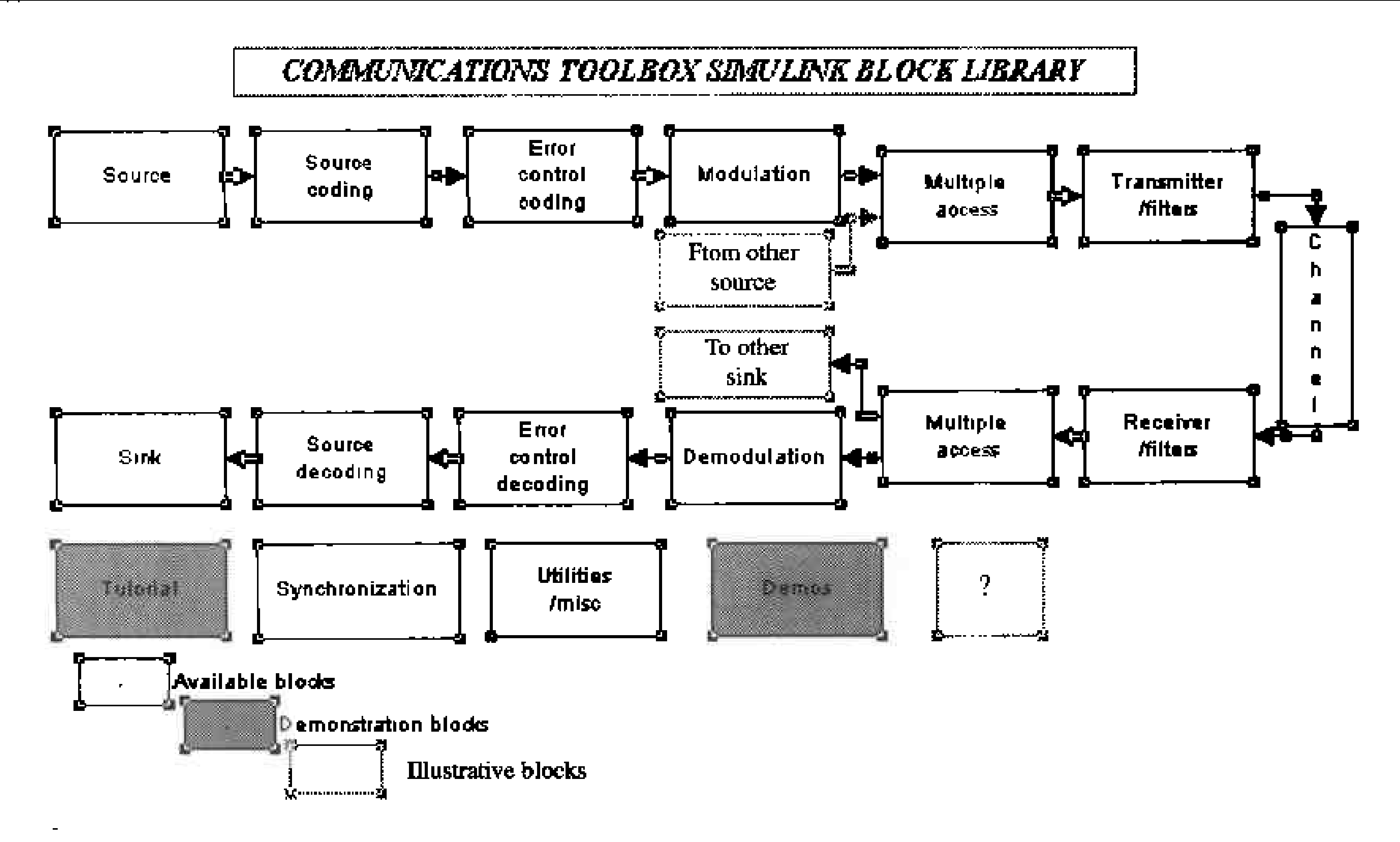


图 9.4　通信系统仿真结构图

附录 A　全书例题索引

表 A.1　全书例题索引

序号	题号	内　容	序号	题号	内　容
1	例 5.1	电阻电路的计算	31	例 6.11	非周期信号(方波)的频谱
2	例 5.2	含受控源的电阻电路	32	例 6.12	用傅里叶变换计算滤波器的响应
3	例 5.3	电阻电路戴维南定理	33	例 6.13	离散信号的 MATLAB 表述
4	例 5.4	一阶动态流电路，三要素公式	34	例 6.14	差分方程的通用递推程序
5	例 5.5	正弦激励的一阶电路	35	例 6.15	求离散系统在各种输入下的响应
6	例 5.6	二阶过阻尼电路的零输入响应	36	例 6.16	二阶巴特沃斯低通数字滤波器的频率响应
7	例 5.7	二阶欠阻尼电路的零输入响应	37	例 6.17	由传递函数转换为零极增益和状态空间模型
8	例 5.8	简单正弦稳态电路	38	例 6.18	由状态空间型转换为传递函数型
9	例 5.9	正弦稳态电路：戴维南定理	39	例 6.19	系统的并联、串联和反馈
10	例 5.10	含受控源的电路：戴维南定理	40	例 6.20	复杂结构的信号流图计算
11	例 5.11	含互感的电路，复功率	41	例 6.21	连续系统状态方程的零输入响应
12	例 5.12	正弦稳态电路，求未知参数	42	例 6.22	离散系统状态方程的响应
13	例 5.13	正弦稳态电路，利用模值求解	43	例 7.1	序列的相加和相乘
14	例 5.14	频率响应：一阶低通电路	44	例 7.2	序列的合成和截取
15	例 5.15	频率响应：二阶低通电路	45	例 7.3	序列的移位和周期延拓
16	例 5.16	频率响应：二阶带通电路	46	例 7.4	离散系统对几种常用信号的响应
17	例 5.17	复杂谐振电路的计算	47	例 7.5	线性性验证
18	例 5.18	网络参数的计算与变换	48	例 7.6	卷积计算
19	例 5.19	匹配网络的计算	49	例 7.7	有限长度序列的 z 变换和逆 z 变换
20	例 5.20	桥梯形全通网络的计算	50	例 7.8	求 z 多项式分式的逆变换
21	例 6.1	连续信号的 MATLAB 描述	51	例 7.9	离散时间傅立叶变换
22	例 6.2	LTI 系统的零输入响应	52	例 7.10	时域采样频率与频谱混叠
23	例 6.3	n 阶 LTI 系统的冲击响应	53	例 7.11	由离散序列恢复模拟信号
24	例 6.4	卷积的计算	54	例 7.12	梳状滤波器零极点和幅频特性
25	例 6.5	LTI 系统的零状态响应	55	例 7.13	低通滤波效果及傅立叶变换时域卷积定理验证
26	例 6.6	系统中有重极点时的计算	56	例 7.14	用 symbolic(符号运算)工具箱解 z 变换问题
27	例 6.7	方波分解为多次正弦波之和	57	例 7.15	基本序列的离散傅立叶变换计算
28	例 6.8	周期信号的频谱	58	例 7.16	验证 N 点 DFT 的物理意义
29	例 6.9	周期信号的滤波	59	例 7.17	验证频域采样和时域采样的对偶性
30	例 6.10	调幅信号通过滤波器。	60	例 7.18	快速卷积

续表

序号	题号	内　容	序号	题号	内　容
61	例 7.19	用 DFT 对连续信号作谱分析	79	例 8.5	连续系统变换为离散系统
62	例 7.20	IIR 滤波器直接型到级联型和并联型转换	80	例 8.6	阻尼系数对二阶系统脉冲响应的影响
63	例 7.21	直接型结构到格型梯形结构转换	81	例 8.7	附加零点对二阶连续系统脉冲响应的影响
64	例 7.22	FIR 滤波器直接型到级联型和格型转换	82	例 8.8	附加极点对二阶连续系统脉冲响应的影响
65	例 7.23	FIR 格型结构到直接型结构转换	83	例 8.9	系统的多种响应曲线
66	例 7.24	用各种窗函数设计 FIR 数字滤波器	84	例 8.10	带时延环节的系统分析
67	例 7.25	用窗函数法设计 FIR 带通滤波器	85	例 8.11	连续和离散系统的根轨迹绘制
68	例 7.26	用 remez 函数设计 FIR 低通滤波器	86	例 8.12	带时延环节的系统根轨迹分析
69	例 7.27	用 remez 函数设计高通滤波器	87	例 8.13	阻尼系数对二阶系统频率响应的影响
70	例 7.28	低通巴特沃斯模拟滤波器设计	88	例 8.14	高阶系统的开闭环频率响应
71	例 7.29	模拟低通转换为数字低通滤波器	89	例 8.15	奈奎斯特曲线及判稳
72	例 7.30	切比雪夫II型低通数字滤波器设计	90	例 8.16	系统的可控性与可观性及其结构分析
73	例 7.31	椭圆带通数字滤波器设计	91	例 8.17	系统的相似变换
74	例 7.32	高通和带通巴特沃斯数字滤波器设计	92	例 8.18	全状态反馈极点配置设计
75	例 8.1	SIMO 系统几种模型转换方法的比较	93	例 8.19	连续系统状态观测器设计
76	例 8.2	含串联和反馈环节的系统传递函数	94	例 8.20	离散系统状态观测器设计
77	例 8.3	用信号流图和 LTI 对象解复杂系统	95	例 8.21	二次型最优调节器的设计
78	例 8.4	复杂框图的结构图变换	96	例 8.22	加权系数对二次型最优调节器的影响

附录 B　MATLAB 基本部分的函数索引

这里列出的是 MATLAB5.1 中基本部分(略去 DEMO)的全部函数名及其所在的库名，带有*号的是 MATLAB 4.x 中没有的；带有**号的是 MATLAB 5.x 中将要取消的。

abs (c)	abcdchk	acos (c)	acosh (c)
acot (c)	acoth (c)	acsc (c)	acsch (c)
addpath(f)	airy*(t)	align(x)	all (n)
angle(c)	ans(d)	any(n)	area* (u)
argnames(e)	asech (c)	asin(c)	asinh(c)
assignin(k)	atan(c)	atan2(c)	atanh(c)
autumn(q)	axes(h)	axis(p)	axlimdlg(j)
balance(m)	bar(u)	bar3 *(u)	bar3h *(u)
barh*(u)	base2dfc*(v)	besselh*(t)	besseli(t)
besselj(t)	besselk(t)	bessely(t)	beta(t)
betacore(t)	betainc(t)	betaln(t)	bicg*(s)
bicgstar*(s)	bin2dec*(v)	bitand*(n)	bitcmp*(n)
bitget*(n)	bitmax*(n)	bitor*(n)	bitset*(n)
bitshift*(n)	bitxor*(n)	blanks (v)	bone(q)
box*(h)	break(k)	brighten(q)	btugroup(x)
builtin(k)	calender(w)	bapture(u)	case*(k)
cat*(d)	cart2pol(t)	cart2sph(t)	caxis(q)
cbedit(x)	cd(f)	cdf2rdf(m)	ceil(c)
cell*(b)	cell2struct*(b)	cellplot*(b)	cgs*(s)
char*(v)	chol(m)	cholinc*(m)	cholest*(m)
cla(h)	clabel(u)	class*(b)	clc(f)
clear(f)	clf(h)	clg**(h)	clock(w)
close(h)	closerq(h)	colmmd(s)	colorbar(q)
colorcube(q)	colormap(q)	colperm(s)	colstyle(u)
comet(u)	comet3(u)	compan(d)	compass(t)
computer(d)	cond(m)	condest(m)	conj(c)
contour(u)	contour3(u)	contourc(u)	contourf*(u)
contrast(q)	conv(a,r)	convhull*(r)	convn*(a)
conv2(a)	cool(q)	copper(q)	copyobj*(h)
corrcoef(a)	cos (c)	cosh(c)	cot(c)
coth(c)	cov(a)	cplxpair(c)	cputime(w)
cross(t)	csc(c)	csch(c)	cumprod(a)
cumsum(a)	cumtrapz*(a)	cylinder(u)	date(w)
datenum*(w)	datestr*(w)	datevec*(w)	datetick*(w)
dbclear(f)	dbcont(f)	dbdown(f)	dblmex*(f)

dbquit(f)	dbstack(f)	dbstatus(f)	dbstep(f)
dbstop(f)	dbtype(f)	dbup(f)	ddeadv(g)
ddeexec(g)	ddeinit(g)	ddepoke(g)	ddereq(g)
ddeterm(g)	ddeunadv(g)	deal*(b)	deblank(v)
dec2bin*(v)	dec2hex(v)	deconv(a，r)	dec2base(v)
delaunay(r)	del2(a)	delete(f)	demo(f)
det(m)	diag(d)	dialog(x)	diary(f)
diff(a)	dir(f)	disp(k)	dlmread(j)
dlmwrite(j)	dmperm(s)	dos(f)	double(b)
dragrect(x)	drawnow(h)	dsearch*(r)	echo(k)
edit(f)	editpath*(f)	eig(m)	eigs(m)
ellipj(t)	ellipk(t)	ellipke(t)	else(k)
elseif(k)	end(k)	eomday*(w)	eps(d)
erf(t)	erfc(t)	erfcx(t)	erfinv(t)
error(k)	errorbar(u)	errordlg(x)	errortrap(k)
etime(w)	etree(s)	etreeplot(s)	eval(k)
evalin*(k)	exist(k)	exp(c)	expin(t)t
expm(m)	expm1(m)	expm2(m)	expm3(m)
eye(d)	ezplot*(e)	factor(t)	fclose()
feather(u)	feof(j)	ferror(j)	feval(k)
fft(a)	fft2	fftn*(a)	fftshift(a)
fgetl(j)	fgets(j)	figure(h)	filepart*(j)
filesep*(j)	fill(u)	fill3(q)	filter(a)
filter2(a)	find(n)	findobj(h)	findstr(v)
finite(n)	fix(c)	flag(q)	flipdim*(d)
fliplr(d)	flipud(d)	floor(c)	flops(d)
fmin(e)	fmins(e)	format(f)	fopen(j)
for(k)	fplot(e)	fprintf(j)	frame2im(u)
fread(j)	frespace(d)	frewind(j)	fscanf(j)
fseek(j)	ftell(j)	full(s)	fullfile*(j)
function(k)	funm(m)	fwrite(j)	fzero(e)
gallery(d)	gamma(t)	gammainc(t) (t)	gammaln(t)
gca(h)	gcd(t)	gcf(h)	gco(h)
gcbo*(h)	gcbf*(h)	get(h)	getenv(f)
getfield*(h)	getframe(u)	ginput()	global
gmres*(s)	gplot(s)	gradient(a)	gray(q)
graymon(h)	grid(p)	griddata(r)	gtext(p)
guide*(x)	hadamard(d)	hankel(d)	help(f)
helpdlg(x)	helpdesk*(f)	helpwin*(f)	hess(m)
hex2dec(v)	hex2num(v)	hidden(q)	hilb(d)
hist(a)	hold(p)	home(f)	hostid(f)
hot(q)	hsv(q)	hsv2rgb(q)	i(d)
if(k)	ifft(a)	ifft2(a)	ifftn*(a)
im2frame*(u)	imag(c)	image(h)	imagesc(h)

此表中各函数名后括号内的字母为库的序号。查某个函数的功能时，可有两种方法。

（1）在本书内粗查：先根据该函数后括号内的小写字母，查表 1.2，找到相应的库名，再根据该库函数表所在的章节、页数，查找该节文字或库函数表内的说明。

（2）在计算机上细查：键入 help 函数名，看其英文说明。

附录 C　信号处理工具箱函数集

表 C.1　信号处理工具箱　版本 4.2(R11)10 Jul 1998

	名　称	功　能	本书中提到的地方
滤波器分析和实现	Fftfilt	搭接相加滤波器实现	7.3 节
	Filter	滤波器实现，滤波器的	7.3 节例 7.4
	Filtfilt	零相位版本	
	Filtic	确定滤波器原始条件	7.4 节
	Freqs	拉普拉斯变换频率响应	7.2 节
	Freqspace	为频率响应设定频率间隔	
	Freqz	z 变换频率响应	7.2 节例 7.12
	Grpdelay	群迟延	
	Impz	脉冲响应(离散的)	7.2 节
	Latcfilt	格形滤波器实现	7.4 节
	Sgolayfilt	Savitzky Golay 滤波器实现	
	Sosfilt	二阶环节(biquad)滤波实现	
	Upfirdn,	高采样，FIR 滤波，再低采样	
	Zplane	画出离散的零极增益点	例 7.12
FIR 滤波器设计	Convmtx	卷积矩阵	
	Cremez	复非线性相位等波动 FIR 滤波器设计	
	fir1	基于窗函数 FIR 滤波器设计(低，高，带通，带阻)	例 7.24
	fir2	基于窗函数的任意响应 FIR 滤波器设计	例 7.24
	Fircls	约束的最小二乘法任意响应滤波器设计	
	Fircls1	约束最小二乘法低通和高通 FIR 滤波器设计	
	Firls	有过渡带，任意响应的 FIR 滤波器设计	
	Firrcos	上升余弦 FIR 滤波器设计	
	Intfilt	插值 FIR 滤波器设计	
	Kaiserord	基于窗函数的 kaiser 滤波器阶数选择	
	Remez	Parks McClellan 最适的 FIR 滤波器设计	7.5 节例 7.26，7.27
	Remezord	Parks McClellan 滤波器阶数选择	7.5 节例 7.26，7.27
	Sgolay	Savitzky Golay FIR 平滑滤波器设计	
IIR 数字的滤波器设计	Butter	巴特沃斯滤波器设计	7.6 节例 7.28
	Cheby1	切贝雪夫-I 型滤波器设计	7.6 节例 7.30
	Cheby2	切贝雪夫-II 型滤波器设计	7.6 节例 7.30
	Ellip	椭圆型滤波器设计	7.6 节例 7.31
	Maxflat	归一化的巴特沃斯低通滤波器设计	
	Yulewalk	耶鲁-沃克滤波器设计	

续表

	名 称	功 能	本节中提到的地方
IIR 滤波器阶数选择	Buttord	巴特沃斯滤波器阶数选择	7.6 节例 7.28
	Cheb1ord	切贝雪夫-I 型滤波阶数选择	7.6 节例 7.30
	Cheb2ord	切贝雪夫-II 型滤波器阶数选择	7.6 节例 7.30
	Ellipord	椭圆型滤波器阶数选择	7.6 节例 7.31
模拟低通滤波器原型	Besselap	贝塞尔滤波器原型	
	Buttap	巴特沃斯滤波器原型	7.6 节
	Cheb1ap	切贝雪夫类型我滤波原型(通频带波动)	7.6 节
	Cheb2ap	切贝雪夫类型 II 滤波器原型(阻带波动)	7.6 节
	Ellipap	椭圆的滤波器原型	7.6 节
频率变换	lp2bp	低通向带通模拟滤波器变换	7.6 节例 7.32
	lp2bs	低通向带阻模拟滤波器变换	7.6 节
	lp2hp	低通向高通模拟滤波器变换	7.6 节例 7.32
	lp2lp	低通向低通模拟滤波器变换	7.6 节
滤波器离散化	Bilinear	有预先修正选项的双线性变换	7.6 节例 7.29，7.32
	Impinvar	脉冲不变性模拟向数字的转换	7.6 节例 7.29
线性系统变换	Latc2tf	格形或者格形梯形向传递函数转换	7.4 节例 7.23
	Residuez	z 变换部分分式展开	7.2 节例 7.8，7.20
	Sos2ss	二阶环节向状态空间转换	7.4 节
	Sos2tf	二阶环节向传递函数转换	7.4 节
	Sos2zp	二阶环节向零极增益转换	7.4 节
	ss2sos	状态空间向二阶环节转换	7.4 节
	ss2tf	状态空间向传递函数转换	7.4 节
	ss2zp	状态空间向零极增益转换	7.4 节
	tf2latc	传递函数向格形或者格形梯形转换	7.4 节例 7.21，7.22
	tf2sos	传递函数向二阶环节转换	7.4 节例 7.20，7.22
	tf2ss	状态空间向传递函数转换	7.4 节
	tf2zp	传递函数向零极增益转换	7.4 节
	zp2sos	零极增益向二阶环节转换	7.4 节
	zp2ss	零极增益向状态空间转换	7.4 节
	zp2tf	零极增益向传递函数转换	7.4 节
窗，函数	Bartlett	巴特利特梨窗函数	
	Blackman	Blackman 窗函数	7.5 节例 7.25
	Boxcar	长方形窗函数	例 7.19
	Chebwin	切贝雪夫窗函数	
	Hamming	hamming 窗函数	例 7.19
	hanning	hanning 窗函数	7.5 节
	kaiser	kaiser 窗函数	
	triang	三角窗函数	

续表

	名　称	功　能	本书中提到的地方
变换	czt	线性调频 z 变换	7.3 节
	dct	离散的余弦变换	7.3 节
	dftmtx	离散的矩阵傅立叶变换	
	fft	快速傅立叶变换	7.3 节例 7.15，7.16
	fftshift	交换矢量的一半	7.3 节
	hilbert	Hilbert 变换	
	idct	离散的逆余弦变换	7.3 节
	ifft	快速傅立叶逆变换	7.3 节例 7.17， 7.18
统计信号处理和谱分析	cohere	相干函数	
	corrcoef	相关系数	
	cov	协方差矩阵	
	csd	互相关谱密度	
	pcov	用协方差方法功率谱估计	
	peig	用特征向量方法功率谱估计	
	pmcov	用修改协方差方法的功率谱估计	
	pburg	用 Burg 方法功率谱估计	
	pmtm	用 Thomson 多带方法功率谱估计	
	pmusic	用 MUSIC 方法的功率谱估计	
	pyulear	用耶鲁-沃克 AR 方法的功率谱估计	
	pwelch	用 Welch 的方法功率谱估计	
	spectrum	Psd，csd，cohere 和 tfe 组合谱	
	tfe	传递函数估计	
	xcorr	互相关函数	
	xcov	协方差函数	
参数建模	arburg	用 Burg 的方法 AR 参数的建模	
	arcov	用协方差方法 AR 参数的建模	
	armcov	用修改协方差方法 AR 参数的建模	
	aryule	用耶鲁沃克方法 AR 参数的建模	
	ident	参看系统辨识工具箱	
	invfreqs	模拟滤波器向频率响应拟合	
	invfreqz	离散的滤波器向频率响应拟合	
	prony	Prony 离散滤波器拟合时间响应	
	stmcb	Steiglitz McBride 迭代的 ARMA 建模	
线性预测	ac2rc	自相关序列向反射系数转换	
	ac2poly	自相关序列向预测多项式安排转换	
	is2rc	反正弦参数向反射系数转换	

续表

	名　称	功　能	本书中提到的地方
线性预测	lar2rc	对数区域的比向反射系数转换	
	levinson	Levinson Durbin 递归	
	lpc	使用自相关方法的线性预测系数	
	lsf2poly	线谱频率向预测多项式的转换	
	poly2ac	预测多项式向自相关序列转换	
	poly2lsf	预测多项式向线谱频率转换	
	poly2rc	预测多项式向反射系数转换	
	rc2ac	反射系数向自相关序列转换	
	rc2is	反射系数向反正弦参数转换	
	rc2lar	反射系数向对数区域比转换	
	rc2poly	反射系数向预测多项式转换	
	rlevinson	逆 Levinson Durbin 递归	
波形产生	chirp	扫频的频率余弦发生器	
	diric	Dirichlet(周期性 sinc)函数	
	gauspuls	高斯的脉冲发生器	
	pulstran	脉冲序列发生器	
	rectpuls	采样的非周期性的方波发生器	
	sawtooth	锯齿函数	
	sinc	sinc 或者(pi*x)/pi*x 函数	例 7.11
	aquare	方波函数	
	tripuls	抽样的非周期性的三角形发生器	
音频支持	sound	重放矢量成为声音	
	soundsc	声音自动定标和重放矢量	
	wavplay	使用视窗音频的输出装置重放声音	
	wavread	读出微软(".wav")声音文件	
	wavrecord	使用视窗音频的输入装置记录声音	
	wavwrite	写微软 (".wav")声音文件	
专门化的运算	cceps	复杂 cepstrum	
	decimate	用低采样速度再抽样	
	deconv	解卷积	
	demod	解调	
	dpss	离散的扩展球形序列(Slepian 序列)	
	dpssclear	从数据库去除离散的扩展球形序列	
	dpssdir	离散的扩展球形序列数据库子目录	
	dpssload	从数据库下载离散的扩展球形序列	
	dpsssave	向数据库写入离散的扩展球形序列	

续表

	名　称	功　能	本书中提到的地方
专门化的运算	interp	用更高采样速度再抽样	
	interp1	通用一维插值透入(MATLAB 实现箱)	
	medfilt1	一维的中值滤波	
	modulate	为通讯仿真调制	
	rceps	实倒谱和最小相位重建	
	resample	用新取样速度再抽样	
	specgram	语言信号谱图	
	spline	立方样条插值	
	vco	压控振荡器	
其他	besself	贝塞尔模拟滤波器设计	
	buffer	把一个矢量缓冲到数据帧的矩阵中去	
	conv2	二维卷积	
	cplxpair	把矢量变为复共轭对	
	fft2	二维快速傅立叶变换	
	ifft2	二维快速傅立叶逆变换	
	polystab	使多项式稳定	
	seqperiod	在一个矢量中找到最小长度重复序列	
	stem	画出离散数据序列	
	strips	画出条形图	
	xcorr2	二维互相关	

附录 D　控制系统工具箱库函数 4.2 版本 (R11)

表 D.1　控制系统工具箱库函数

功能类型	命　　令	功　　能	本书中介绍的章节
LTI 模型建立	tf	建立或转换成传递函数模型	8.1 节，例 8.6
	zpk	建立或转换成零极增益模型	8.1 节，例 8.14
	ss	建立或转换成空间模型	8.1 节，例 8.9, 8.16
	dss	建立描述状态空间模型	8.1 节
	frd	建立或转换成频率响应数据模型	
	filt	建立数字的滤波器模型	8.1 节
	set	置定/修改 LTI 模型属性的命令	8.1 节
	ltimodels	各类型的 LTI 模型的详细帮助	
	ltiprops	LTI 数据提取可用工具详细帮助	
数据提取	tfdata	提取传递函数分子和分母数据	8.1 节
	zpkdata	提取零极增益数据	8.1 节
	ssdata	提取状态空间矩阵	8.1 节
	dssdata	提取描述状态空间矩阵	8.1 节
	frdata	提取频率响应数据	
	get	提取 LTI 的属性值接近的机会价值观做模特儿	8.1 节
模型尺度和特征	class	模型类型(tf、zpk、ss 或者 frd)	8.1 节
	isa	测试 LTI 模型是否为给定类型	8.1 节
	size	模型大小和阶次	
	ndims	维数	
	isempty	LTI 模型是空时为真	
	isct	是连续时间模型时为真	8.1 节
	isd	为离散时间模型时为真	8.1 节
	isproper	为恰当 LTI 模型时为真	
	issiso	为 SISO 模型时为真	8.1 节
	reshape	使 LTI 模型的阵列再成形转换	
转换成转换函数	tf，zpk，ss，frd	见 LTI 模型建立函数栏	8.1 节
	chgunits	改变 FRD 模型频率点的单位	
	c2d	连续到离散时间的转换.Con	8.1 节，例 8.5，8.6
	d2c	离散到连续时间的转换	8.1 节，例 8.5
	d2d	再抽样	8.1 节，例 8.5

续表

功能类型	命　令	功　能	本书中介绍的章节
扩展运算符	+ 和 −	LTI 系统加减（并联）	8.1 节，例 8.2，8.3
	*	LTI 系统相乘（串联）	8.1 节，例 8.2，8.3
	\	LTI 系统左除，sys1\sys2 =inv(sys1)*sys2	8.1 节，例 8.2，8.3
	/	LTI 系统右除，sys1/sys2 = sys2*inv(sys2)	8.1 节
	^	LTI 系统乘幂	8.1 节
	'	共轭转置	8.1 节
	.'	输入/输出映射转置	8.1 节
	[..]	LTI 系统沿输入或输出拼接	
	stack	把 LTI 系统/阵列沿阵列的某些维放入堆栈	
	inv	LTI 系统求逆	8.1 节，例 8.2，8.3
模型动态性能分析	pole，eig	系统极点	8.2 节，例 8.20
	zero	系统(传送)零点	8.2 节
	pzmap	零极点图	8.2 节，例 8.8，8.9
	dcgain	连续低频(D.C)增益	8.2 节
	norm	LTI 系统的范数	8.2 节
	covar	对白噪声的连续协方差响应	8.2 节，例 8.8
	damp	系统极点的阻尼系数和固有频率	8.2 节，例 8.10，8.14
	esort	按实部从大到小对连续极点进行排序	8.2 节
	dsort	从大到小对离散极点进行排序	8.2 节
时延	hasdelay	是有时延的系统时为真	
	totaldelay	在每一对输入/输出之间总时延	
	delayfr	用在 z=0 处的极点或者 FRD 的相移代替时延	
	pade	用 Pade 多项式近似时延	8.1 节，例 8.10，8.12
状态空间模型	rss，rmodel	随机生成稳定的连续状态空间模型	8.1 节，例 8.9
	drss，drmodel	随机稳定的离散状态空间模型	8.1 节
	ord2	生成二阶系统的状态方程或传递函数	8.1 节，例 8.6，8.13
	ss2ss	状态空间的相似变换	8.4 节
	canon	系统转换为标准形式	8.4，例 8.17
	ctrb，obsv	可控性阵，可观性阵	8.4，例 8.16
	gram	可控性和可观性 gramians 阵(用于时变系统)	8.4 节
	ssbal	状态空间实现的对角线平衡	8.4 节
	balreal	平衡实现.Gramian 输入/输出余额	8.4 节
	modred	模型状态降阶	8.4 节
	minreal	最小实现和零极点对消	8.4 节
	sminreal	结构极小实现	8.4 节
时间响应	ltiview	响应分析图形界面（LTI 观测器，及频率响应分析）	
	step	阶跃响应	8.2 节，例 8.8，8.10
	Impulse	冲击响应	8.2 节，例 8.6，8.10
	initial	连续状态空间系统的初始状态响应	8.2 节，例 8.9
	lsim	连续系统给定输入的输出计算	8.2 节，例 8.9
	gensig	为 LSIM 产生输入信号	8.2 节
	stepfun	产生单位阶跃输入	8.2 节

续表

功能类型	命令	功能	本书中介绍的章节
频率响应	bode	频率响应的波德曲线	8.3节，例8.13，8.14
	sigma	连续奇异值频率曲线	8.3节
	nyquist	奈奎斯特曲线	8.3节，例8.15
	ngrid	画Nichols图网格	8.3节
	nichols	Nichols图	8.3节
	margin	边缘增益和相位裕量	8.3节，例8.14，8.15
	freqresp	在一频率格栅上的频率响应	8.3节
	evalfr	在给定频率上求频率响应的数值	8.3节
系统互相连接	append	通过添加输入和输出把两LTI系统合成	8.1节，例8.4
	parallel	系统并联后的等效系统	8.1节，例8.3
	series	系统串联后的等效系统	8.1节，例8.3
	feedback	系统反馈连接后的等效系统	8.1节，例8.3，8.10，8.12
	lft	广义的反馈连接	
	connect	从方块图导出状态空间模型	8.1节，例8.4
根平面分析设计工具	rltool	根轨迹设计GUI	8.2节
	rlocus	伊文斯根轨迹	8.2节，例8.11，8.12
	rlocfind	用人机交互确定根	8.2节，例8.11，8.12
	sgrid	画s平面根网格线	8.2节，例8.11，8.12
	zgrid	画z平面根网格线	8.2节，例8.11，8.12
	acker	单输入单输出系统极点配置	8.5节，例8.18，8.19，8.20
	place	MIMO系统极点配置	8.5节，例8.18，例8.19，8.20
	estim	给定增益构成估计器	8.5节，例8.19
LQG设计工具	lqr	线性平方调节器设计	8.5节，例8.21，8.22
	dlqr	离散线性平方调节器设计	8.5节
	lqry	输出加权的调节器设计	8.5节
	lqrd	用连续代价函数的离散调节器设计	8.5节
	kalman	离散Kalman估计器	8.5节
	kalmd	被控对象的离散Kalman估计器	8.5节
	lqgreg	给定LQ增益和Kalman估计器构成LQG调节器	8.5节
	augstate	通过添加状态变量扩大输出	
矩阵方程求解	lyap	解连续Lyapunov方程式	8.5节
	dlyap	解离散李雅普诺夫方程	8.5节
	care	解代数黎卡提方程	8.5节
	dare	解离散代数Riccati方程式	8.5节
演示	ctrldemo	对控制系统工具箱的介绍	
	jetdemo	喷气式运输飞机偏荡阻尼的经典设计	
	diskdemo	硬盘驱动器数字控制器的设计	
	milldemo	对轧钢厂的SISO和MIMO LQG控制	
	kalmdemo	Kalman过滤器设计和仿真	

续表

功能类型	命　　令	功　　能	本书中介绍的章节
对象方法库和其他目录	@lti	lti 对象的方法库目录	8.1 节
	@ss	ss 对象的方法库目录	8.1 节
	@tf	tf 对象的方法库目录	8.1 节
	@zpk	zpk 对象的方法库目录	8.1 节
	obsolete	新版本中将要废除的函数目录	

由于引进了 LTI 对象的概念，控制工具箱的新旧版本差别很大，原有的控制工具箱中有很多函数，特别是那些连续与采样系统分别设立的函数，现在都可以合成一个。因此，很多函数已经废除。在新旧交替的时候，读者若照两年前出版的 MATLAB 书籍来编程序，在新版本 MATLAB 环境下就不能运行。因此把 Mathworks 公司提供的废除的函数名列成下表，供读者编程时参考。

表 D.2　老版本控制工具箱中将要被废除的函数 obsolete

are	ddamp	dlqew	Dstep	lqe	plotnic	timvec
blkbuild	ddcgain	dlqry	Dtimvec	lqe2	plotnyq	tzero2
c2dm	destim	dlsim	Exresp	lqed	poly2str	
c2dt	dexresp	dmodred	Fbode	lqew	printmat	
chop	dfrqint	dmulresp	Freqint	lqr2	printsys	
cloop	dfrqint2	dnichols	Freqint2	lyap2	ric	
d2cm	dgram	dnyquist	Givens	miniplot	schord	
dbalreal	dimpulse	dreg	Gram	mulresp	ssdelete	
dbode	dinitial	dric	Imargin	perpxy	ssselect	
dcovar	dlqe	dsigma	lab2str	plotbode	subaxes	

参 考 文 献

[1] 陈怀琛. MATLAB 及其在理工课程中的应用指南. 西安：西安电子科技大学出版社，2000

[2] 邱关源. 电路（第四版）. 北京：北京高等教育出版社，1999

[3] 吴大正，王松林，王玉华. 电路基础（第二版）. 西安：西安电子科技大学出版社，2000

[4] 郑君里，应启珩，杨为理. 信号与系统（第二版）. 北京：北京高等教育出版社，2000

[5] 吴大正，杨林耀，张永瑞. 信号与线性系统分析（第三版）. 西安：西安电子科技大学出版社，1998

[6] 丁玉美，高西全编著. 数字信号处理（第二版）. 西安：西安电子科技大学出版社，2001

[7] 陈怀琛，王朝英，高西全等译. Ingle V K, Proakis J G 著. 数字信号处理及其 MATLAB 实现. 北京：电子工业出版社，1998

[8] 楼顺天，李博菡. 基于 MATLAB 的系统分析与设计—信号处理.西安：西安电子科技大学出版社，1998

[9] 陈怀琛，黄道君. 控制系统 CAD 及 MATLAB 语言. 北京：电子工业出版社，1996

[10] 徐昕，李涛，伯晓晨等. MATLAB 工具箱应用指南—控制工程篇. 北京：电子工业出版社，2000

[11] 欧阳黎明等. MATLAB 控制系统设计—工程师工具软件应用系列. 北京：国防工业出版社，2000

[12] 楼顺天，于卫. 基于 MATLAB 的系统分析与设计—自动控制. 西安：西安电子科技大学出版社，1998

[13] Buck J R，Daniel M M. Singer A C. Computer Explorations in Signals and Systems Using MATLAB. Prentice Hall,1997

[14] MATLAB User's Guide. The Mathworks Inc，1998

[15] Duane Hanselman，Bruce R. Littlefield，Mastering MATLAB 6： A Comprehensive Tutorial and Reference,Prentice Hall PTR，12/2000

[16] William J. Palm，Introduction to MATLAB 6 for Engineers，McGraw-Hill Higher Education，02/2001

[17] 胡光锐. 数字信号处理——理论、算法与实现. 北京：清华大学出版社，1977

［Ｇｅｎｅｒａｌ　Ｉｎｆｏｒｍａｔｉｏｎ］
书名＝ＭＡＴＬＡＢ及在电子信息课程中的应用
作者＝
页数＝３１１
ＳＳ号＝０
出版日期＝